

Machine Learning, Data Mining, and the World Wide Web: Design of Special-Purpose Search Engines

Andries F. Kruger



Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Science
at the University of Stellenbosch

Supervisor: Prof Christian W Omlin

April 2003

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature:

Date:

Abstract

We present DEADLINER, a special-purpose search engine that indexes conference and workshop announcements, and which extracts a range of academic information from the Web. SVMs provide an efficient and highly accurate mechanism for obtaining relevant web documents. DEADLINER currently extracts speakers, locations (e.g. countries), dates, paper submission (and other) deadlines, topics, program committees, abstracts, and affiliations. Complex and detailed searches are possible on these fields. The niche search engine was constructed by employing a methodology for rapid implementation of specialised search engines. Bayesian integration of simple extractors provides this methodology, that avoids complex hand-tuned text extraction methods. The simple extractors exploit loose formatting and keyword conventions. The Bayesian framework further produces a search engine where each user can control each fields false alarm rate in an intuitive and rigorous fashion, thus providing easy-to-use metadata.

Oorsig

Ons stel DEADLINER bekend: 'n soekmasjien wat konferensie en werkvergaderingsaankondigings katalogiseer en wat uiteindelik 'n wye reeks akademiese byeenkomsmateriaal sal monitor en onttrek uit die Web. DEADLINER herken en onttrek tans sprekers, plekke (bv. landname), datums, o.a. sperdatums vir die inlewering van akademiese verrigtings, onderwerpe, programkomiteë, oorsigte of opsommings, en affiliasies. 'n Grondige soek is moontlik oor en deur hierdie velde. Die nissoekmasjien is gebou deur gebruik te maak van 'n metodologie vir die vinnige oprigting van spesialiteitsoekmasjiene. Die metodologie vermy komplekse instelling m.b.v. hande-arbeid van die teksuittreksels deur gebruik te maak van Bayesiese integrering van eenvoudige ontsluiters. Die ontsluiters buit dan styl- en gewoonte-sleutelwoorde uit. Die Bayesiese raamwerk skep hierdeur 'n soekmasjien wat gebruikers toelaat om elke veld se kans om verkeerd te kies op 'n intuïtiewe en deeglike manier te beheer.

Acknowledgements

I am grateful to Prof. C. Lee Giles who gave me the unique and pleasurable opportunity of working with him at NEC Research Institute, Princeton (USA). I would also like to thank his colleagues Frans Coetzee, Gary Flake, Eric Glover and Steve Lawrence for their collaboration, and very good ideas. I thank my advisor, Prof. Christian W. Omlin, who organised my internship at NECI, and for his support and guidance in this thesis. The following people helped me keep my sanity: Walter “Wallie” Andrag, Thomas Bitzer, Ben Bredenkamp, Izak Burger, Tomas By, James Connan, Sandip Debnath, Michelangelo Diligenti, Steve Kroon, Andre Muller, Finn Aarop Nielsen, David “Dave” Pennock, Kenji Satoh, Sean “Gadgets” Snyders, Jacobus van Zyl, Dane Walsh, Ting Wang, Brian Whitman, and Bin Yu.

to the Creator, and my parents

Contents

1	Introduction	1
1.1	Motivation	1
1.2	A Historical Perspective	3
1.3	Internet Computing: The Paradigm of the 21st Century?	5
1.4	Problem Statement	7
1.5	Objectives	8
1.6	Methodology	9
1.7	Accomplishments	10
1.8	Thesis Outline	11
2	Data Mining on the World Wide Web	13
2.1	Information Retrieval on the World Wide Web	13
2.2	Web Navigation	15
2.3	Web Crawlers	16
2.4	Focused Web Crawlers	17
2.4.1	ARACHNID: Adaptive Retrieval Agents Choosing Heuristic Neighbour- hoods for Information Discovery	18
2.4.2	Using Reinforcement Learning to Spider the Web Efficiently	20
2.4.3	Focused Crawling: A New Approach to Topic-Specific Web Resource Dis- covery	22
2.4.4	Focused Crawling Using Context Graphs	24
2.5	General Purpose Search Engines	26
2.6	Metasearch Engines	27
2.7	Special Purpose Search Engines	29
2.8	Opportunities for Machine Learning	31
2.9	Related Work	32
2.10	Summary	33

3	Document Classifiers	34
3.1	Introduction	34
3.2	Bayesian Learning	34
3.3	Neural Networks	36
3.4	k -Nearest Neighbour	37
3.5	Support Vector Machines	37
3.5.1	Separable Patterns and Building a Support Vector Machine	37
3.5.2	Non-separable Patterns and Support Vector Machines	40
3.5.3	Building a Support Vector Machine	41
3.6	Information Extraction	43
3.7	A Case for a Hybrid Approach	44
4	ROCs and Detection/Decision Theory	45
4.1	Introduction	45
4.2	Bayes Criterion	46
4.3	Neyman-Pearson	47
5	DEADLINER: Building a New Niche Search Engine	51
5.1	Introduction	51
5.2	General Architecture Overview	54
5.2.1	Stage I: Document Retrieval	56
5.2.2	Stage II: Pre-Screening Using SVMs	56
5.2.3	Stage III: Bayesian Detector Fusion	57
5.2.4	Stage IV: Presentation and Cataloguing	60
5.3	Filters used in DEADLINER	62
5.3.1	Features and Filters	62
5.3.2	Heuristics	63
5.4	Performance	66
5.5	Summary	71

6	Conclusions and Directions for Future Research	72
6.1	Conclusions	72
6.2	Directions for Future Research	73
A	Bibliography	74

1 Introduction

1.1 Motivation

Since the inception of the World Wide Web, a multitude of information frontiers with many, and sometimes surprising, advantages have become apparent. The ever-increasing demand on more and higher quality information presents unique challenges in the information dissemination environment, and with the advent of massive personal and commercial involvement in the Web, significant technical improvements are required. Advances in hardware necessitate advances in software, especially since the Web is mostly an abstract software concept. The Web is a collection of information artifacts, which can form uni-directional links to other information artifacts.

Thus, the Web forms a graph structure, with no theoretical limit on the number of uni-directional links (edges) from artifacts (nodes) to other artifacts (nodes), and no theoretical limit to the size or number of artifacts. There is also no constraint on where these artifacts are stored. Storage devices range from personal computers to large mainframes, with the connecting hardware and software network forming the Internet¹.

Questions that arise from this environment include location, classification, indexing, extraction, and dissemination of information.

The average time after which a web page disappears is estimated at 75 days by Alexa². AltaVista reported in their bow tie experiment (see also Section 2.4) that the Internet forms a core of pages that are tightly linked together, with some pages linked to from the core, some pages linking only to the core, some pages neither linking to, nor linked to by the core, forming a bow tie with four components. Lawrence and Giles reported the estimated mean age of 186 days (median 57) of web documents indexed by some of the bigger general purpose search engines. They report that the biggest coverage of the Web by a (meta-)search engine is 42%. Combining these facts with the

¹We note that the difference between the Internet and the World Wide Web is that the Internet is the collection of communication protocols, such as TCP, UDP, etc. over a hardware communication network, whereas the World Wide Web uses only one protocol (Hypertext Transfer Protocol or HTTP), which runs on top of the Internet protocols.

²<http://www.alexa.com>

explosive growth in the size of the Web (see Section 2.1), we find a bleak picture of the currency and availability of information, and the techniques used for *locating* information on the web [1, 2].

If we suppose that we actually can obtain information, we still do not know whether the information is relevant to our information need. *Classification* in some form, either before or after querying an information source, can greatly reduce the amount of work and thus the time spent searching for relevant material. Traditionally, information in a conventional library is categorised in a systematic manner, and this approach is used with some search engines. Can this be done automatically, or is Yahoo's³ manually constructed topic hierarchy the only way? Do we actually need information artifacts classified into a taxonomy to be able to efficiently locate relevant information? These taxonomies require a massive amount of effort to create. Additionally, the artifacts they contain are often not relevant to the category, since the artifacts change as web page authors' interests change. Automatic categorisation seems a good solution, but no automatic classification system can handle the thousands of categories, and provide a very good classification. In contrast, there are some very good classifiers for the binary, or two-class case (see e.g. Section 3). How can we use this classification technology to obtain relevant information? We believe that the application of classifiers in special-purpose search engines works towards this goal [3, 4].

On the one hand, indexing information on the scale of the Web is a technically hard problem. Few companies and individuals have the resources to build an index of the Web: in fact few companies can even crawl the whole Web⁴. Building a reverse index requires significant computing resources. Similar to our argument above, large indexes are also very quickly out of date. On the other hand, many special-purpose search engines do not require massive resources, and can be maintained by an individual or a small company. It is therefore reasonable to investigate methods for building these types of search engine. Maintaining an index of a subset of the Web is much easier, and, since the index is smaller, we do not need to use some of the more advanced indexing techniques. Additionally, storage devices are cheap. Maintaining current information artifacts is consequently

³<http://www.yahoo.com>

⁴Alexa (<http://www.alexa.com>) donated a snapshot of the Web to the US Library of Congress — a feat not many companies can claim.

much easier, since the artifacts can be revisited frequently. The index can be comparatively large, and we can add *metadata* we otherwise would not be able to.

Except for some very general metadata⁵, extracting information from web artifacts is generally not feasible for general-purpose search engines, because the computational complexity requires significant computing resources. Why do we need metadata? Metadata can increase relevancy and accuracy of information. We can answer structured questions, e.g. what is the average cost of red cars in Tahiti, which is problematic with general-purpose search engines. Constructing methods for extracting metadata is highly specific to the type of artifact. For example, obtaining dates from images and text is very different in nature, again motivating investigations into special-purpose search engines.

How do we disseminate information after extracting metadata? How do we choose a search engine that will support our information need? Some of these questions are being explored in metasearch engine research, but very little work is being done in integrating results from different special-purpose search engines.

Finally, the large number of special-purpose search engines on the Web supports our argument for investigating special-purpose search engines, and the tools for their efficient construction.

1.2 A Historical Perspective

Communication of ideas has always been pivotal to the success of major human endeavours. The Internet and the World Wide Web are significant achievements of the late 20th century, and similar to the industrial revolution, are bringing about changes to society that were almost impossible to envision a few decades ago. Today, it is possible to work, play, and even in some sense visit, people all over the world, without moving from the comfort of your home or office. In this section, we describe some of the earliest contributors that set in motion this information revolution. First we give a short overview of the history of the Internet, followed by a history of the World Wide Web

⁵See PageRank, Sections 2.2 and 2.3.

(WWW or Web for short).

In August 1962 J.C.R. Licklider envisioned a “Galactic Network”, similar to the Internet we have today, where computers are interconnected globally. Licklider, the first head of Defence Advanced Research Projects Agency’s (DARPA) research program, convinced his successors of the importance of the “Galactic Network”. One of his successors, Lawrence G. Roberts, was convinced of the theoretical feasibility of packet switching by Leonard Kleinrock. In 1965, Roberts connected the TX-2 computer in Massachusetts via a phone line to the Q-32 in California, providing an experimental proof of concept⁶. In 1967, Roberts published his plan for the Advanced Research Projects Agency Network, or more commonly known as ARPANET. Roberts consulted with his peers, and ARPANET was refined until August 1968, with a final line speed of 50 kbps (kilobits per second). In December 1968, the group Bolt, Beranek and Newman (BBN) headed by Frank Heart won the contract for quotations for ARPANET. The first node was built at UCLA, home of Kleinrock’s Network Measurement Center in September 1969. “Augmentation of Human Intelligence”, Doug Engelbart’s project at the Stanford Research Institute, formed the second node. And, in October 1969, the first host-to-host message was sent. By the end of 1969, UC Santa Barbara and the University of Utah were linked in for a total of four nodes. The Internet was born. The Network Working Group under S. Crocker finished the initial host-to-host protocol in December 1970. This protocol was implemented in 1971–1972, and following its successful public demonstration in October 1972, email was introduced [5].

ARPANET developed into the Internet as we know it today (defined in Section 1.3), and the technical approach used is known as “open architecture networking”. This approach rests on the idea that, rather than restricting all networks to the same underlying technology, an inter networking architecture is defined. The open architecture was first defined by Robert E. Kahn, who wished to use radio for data transmission. The Network Control Protocol (NCP) used on ARPANET could not handle errors in transmission. Kahn decided to develop a new protocol, and he assumed (1) that each network is stand-alone, and no internal changes are required for connection to the Internet, (2) communication is on a best effort basis, (3) “black boxes” connect the networks (later called

⁶This is the first wide area network (WAN) ever built.

gateways and routers), and (4) there is no global control at the operations level. The new protocol Kahn and Vint Cerf developed is called Transmission Control Protocol/Internet Protocol (TCP/IP). An interesting point that shows the level of cooperation possible in the academic world, is the transition from NCP to TCP/IP. Several years of planning went into the transition, and this was done globally on 1 January 1983. ARPANET was decommissioned in 1990, and TCP/IP replaced most other network protocols worldwide [5, 6].

Libraries, newspapers and word of mouth, have always served as the *de facto* methods for dissemination of information. The World Wide Web has changed the way information is stored and accessed. Interestingly, the idea to link documents together on a computer had already been proposed as early as 1945 by Vannevar Bush, on a machine called a Memex. In 1960, Doug Engelbart prototyped a system that handled email and allowed hypertext browsing⁷. However, thirty years passed before work started on an information management system called the World Wide Web. Mosaic, the first browser that came into wide public use, was released in 1993 for the X windowing system, the Macintosh and the IBM PC. Since then, the Web has grown exponentially, and already contains more than a billion documents and other artifacts of various degrees of utility. Information has never been this abundant and available — and never this difficult to obtain [7].

1.3 Internet Computing: The Paradigm of the 21st Century?

What is the Internet? On 24 October 1995, the following resolution was passed:

RESOLUTION: The Federal Networking Council (FNC) agrees that the following language reflects our definition of the term “Internet”. “Internet” refers to the global information system that – (i) is logically linked together by a globally unique address space based on the Internet Protocol (IP) or its subsequent extensions/follow-ons; (ii) is able to support communications using the Transmission Control Protocol/Internet Protocol (TCP/IP) suite or its subsequent extensions/follow-ons, and/or other IP-compatible

⁷He also invented the mouse in the process.

protocols; and (iii) provides, uses or makes accessible, either publicly or privately, high level services layered on the communications and related infrastructure described herein. [5, 6]

Internet computing⁸ can then be viewed as the application of computing methods and algorithms to the novel Internet environment. The Internet is about the distribution of information in order to facilitate communication. Analysing the communications found on the Internet can yield anything from commercial applications to a theoretical understanding of human interaction. Research on these communications can yield optimised models in many areas. One interesting area that comes to mind is large project management⁹. The efficient communication provided by the Internet allows projects to be finished more quickly, and many people from remote locations can cooperate on the same project. The rapid exchange of ideas pushes many projects forward, since it is more efficient to read a considered email than to talk with many people.

The increasing commercialisation of the Internet, with an estimated revenue of 377 billion US dollars for 2000, and a projected 717 billion for 2001, points to the expected increase in the number of web users. Estimates of the number of people using the Internet currently ranges from 391 to 407 million users, which is expected to double within the next 3 to 4 years. People present opportunities, ranging from friends and leisure to economics. New Internet related technologies are invented every day, and to handle the e-commerce, one could envision an explosive growth in Internet size. But what of the rest of this century? It is impossible to look too far into the future; however, the Internet is already having a significant influence on the world economy, with many business advertising on the Web, and with business-to-business transactions becoming more commonplace. Internet computing promises to change many aspects of our lives, from the way we shop to the way we are entertained. In the foreseeable future, bandwidth should be cheap enough to allow real time down loading of movies. This could change your computer into a television, or, if you connect a video projector to it, into a home cinema, with the movie of your choice. It is

⁸The theory of computing was originally proposed by Alan Turing in 1936, with the aim of investigating algorithmic processes with a computational model. The model he developed is the basis on which every computer today works [8].

⁹The construction of the Linux operating system is a prime example.

already possible to see a change in advertising — many companies advertise with the high-traffic Internet companies, instead of advertising in newspapers or on the local television [9, 10, 11].

Distributed wide area network computation is another facet that can increase quality of life by providing access to computationally intensive applications. The average user might not have the resources to compute Mersenne primes¹⁰ or run complex information extraction algorithms in real time. Distributed Internet computation can be motivated by the millions of computers that are idle for a large part of the day, since it is night/evening in approximately half of the world at any specific time.

Where are we going from here? Our imaginations are our only limit.

1.4 Problem Statement

What is a search engine? A *search engine* is a software construction that facilitates dissemination of information by providing an index on information artifacts that simplifies the location of relevant information. The information requirement is expressed as a *query*, typically expressed using a Boolean query language. *General purpose search engines* over text and web documents usually index on terms (words, phrases), and sometimes on hyper links as well. *Metasearch engines* query other search engines, and present the integrated results to the Web user. *Special-purpose search engines* index on domain specific data and metadata over domain relevant information artifacts.

The Internet has seen a profusion of special-purpose search engines, ranging from text to multimedia. However, constructing a special-purpose search engine is domain dependent, which adds manual labour to construction. Reducing this labour to a minimum would benefit the Internet community as a whole. Developing appropriate tools could facilitate solutions towards this goal. We believe that this thesis is a step in that direction, where we construct a special-purpose search engine as a case study aimed at finding relevant tools and developing a methodology that enables us to obtain document titles, speakers, locations, dates, paper submission (and other) deadlines, topics,

¹⁰Mersenne primes are prime numbers of the form $2^n - 1$ for n some prime. See also <http://www.mersenne.org>.

program committees, abstracts and affiliations for conference and workshop announcements.

Common elements of software projects include requirements, design, architecture, implementation, testing, and maintenance. We concentrate in this thesis on method and architecture, and not on the other software engineering issues.

To engineer a special-purpose search engine, the required functionality includes the presentation and dissemination of information, after locating, extracting, and indexing on relevant segments of relevant information artifacts. Presentation is usually domain specific, and not a primary focus of this thesis. Obtaining relevant artifacts can involve classification, web crawling, query modified web search, polling and web user contributions. Information extraction is domain dependent, and techniques appropriate to a specific domain need to be developed. However, except for vast differences, such as textures in images and some text special-purpose search engines, information extraction techniques are adaptable to different domains, with only minor customisation. Information presentation, dissemination, and indexing, though important and necessary, are appurtenant to our primary goals.

We will propose an architecture that supports the above constraints. Ideally, this architecture is general, and adaptable to new special-purpose search engines with no or little modification. The components of the architecture are modular and loosely coupled as far as possible.

Exploring existing technologies can yield appropriate solutions, and this thesis explores some of the related research.

1.5 Objectives

A special-purpose search engine needs some way to locate relevant documents. Since the search engine is special-purpose, and therefore only needs to index a small subset of the Web, a focused crawler is appropriate. Documents can be acquired via a category-specific search engine, such as the engine developed by Glover *et al.*, and then generating a query based on extracted information. We present efficient methods for locating artifacts in Section 2 [12, 13].

Once we have obtained some web document we need to determine the document's relevance more accurately, since, even though the methods used confer some degree of relevancy to located documents, there will be noise. To increase the accuracy of our relevancy judgement, we need a high quality classifier trained on relevant documents.

Special-purpose search engines are built on the premise that, by constraining the underlying domain, we can utilise domain knowledge. This premise is the real power behind special-purpose search engines, since common features can be exploited. Extraction of these features allows complex queries. The information sources for a specific domain can be diverse in location and form. Modelling the domain features is the big obstacle in constructing the search engine. Creating a general methodology for modelling the domain so that we can locate and extract information from the information artifacts is the primary goal of our methodology, combined with locating the artifacts themselves. Modelling the domain for document classification is described in Section 5. We model domain dependent metadata with a set of simple, partial filters that are based on regular expressions. These filters are relatively easy to specify, but to combine them appropriately is challenging, and is a major focus of this thesis.

Secondary goals include building a reverse index over the extracted metadata and the documents, and presentation of the extracted data in a clear user interface.

1.6 Methodology

Based on the objectives in Section 1.5, we develop an architecture that supports the functionality described. Utilising this architecture, we choose and refine, or develop components that achieve the stated goals.

To locate artifacts we notice that, since the special-purpose search engine indexes only a small portion of the Web, we can use techniques available to normal web users to procure relevant artifacts. One such approach, focused crawlers, is presented in Section 2.4. Other approaches include query modified search and metasearch.

We use both a focused crawler and query modified metasearch, which can result in greater diversity of artifacts. The yields from the above approaches are then filtered through a high quality filter to improve the quality of the documents, because irrelevant artifacts could further skew the distribution of relevant information in the database.

Extracting information from web pages comprises at least two aspects: locating a relevant portion, and then extricating the metadata. Constructing partial filters is a relatively easy task. We aim at combining the location, extraction and the partial filters in a rigorous way. Bayesian integration provides a methodology that allows to do this; it determines the text block that best fit the training data, and then applies the filters or some overriding heuristic to extract metadata.

Having extracted metadata, we need a reverse index on the extracted data. Since the metadata is structured, we used an SQL (structured query language) database to store the data. SQL databases do not provide an inherent reverse index, and since hashes provide quick access to reverse indexed data, we chose the Berkeley DB in combination with the freeware SQL server MySQL.

We base the interface for presentation and dissemination on the database structure, starting with the title of an artifact. This provides a star shaped interface starting at the main page. Other configurations are certainly possible, and may be even desirable; we will leave most of this component for Section 6.

1.7 Accomplishments

We develop an architecture for constructing a special-purpose search engine that supports location of artifacts, a general methodology for extraction of metadata from artifacts, indexing over documents, and an interface for presentation of information on conference and workshop announcements.

After locating artifacts using query-modified search and focused crawling, we pre-screen the artifacts using a support vector machine (SVM), yielding a high quality subset of the Web. A special-purpose search engine's domain of operation is constrained, with common elements in the informa-

tion artifacts. This results in a general methodology for integrating multiple simple detectors. We demonstrate this methodology for building niche search engines by constructing DEADLINER. DEADLINER is a niche search engine that catalogues conference and workshop announcements. We use this methodology to classify text blocks and extract metadata, such as titles, speakers, etc.¹¹ found in the announcements. Users can perform detailed searches on these fields, and they are also allowed to set the false alarm rate, i.e. the probability of making a wrong decision, easily and intuitively [13, 14].

Finally, after storing and index the resulting metadata, the results of queries are presented to users.

1.8 Thesis Outline

We have provided an introduction to the thesis in the previous sections. We will proceed with a short overview of information retrieval in Section 2, where we highlight some of the characteristics and issues faced by information retrieval on the Web. Some results on web navigation are presented, where the importance of an artifact can be estimated by analysing the link structure of the Web. We give an overview of web crawlers, and detail focused crawlers, where we discuss some of the results in the literature, ranging from a genetic algorithm approach with ARACHNID through Q-learning and naive Bayes, and we finish with a recent approach that integrates constructed context graphs of small subsets of the Web. On the one hand, the small number of general-purpose search engines might be an indication of the resources required to build one; on the other hand, we have metasearch engines, which need far fewer resources. We give a brief motivation for special-purpose search engines, and provide a few examples. We discuss opportunities for machine learning. Lastly, we highlight some details on ResearchIndex and CORA, the most prominent special-purpose search engines in the literature.

Document classifiers are discussed in Section 3. We provide the basics for some of the more well-known classifiers. Application of Bayes' theorem to text classification, a neural network approach, and k -nearest neighbour classifiers are discussed. We provide a brief introduction to support vector

¹¹See Section 5.1 for a more complete list

machines, starting with the case for separable patterns. We subsequently give an overview of non-separable patterns, and finish the section by building an SVM.

DEADLINER is discussed in Section 5. We provide a general architecture overview describing document retrieval and classification. The filters we use to classify and extract metadata and the heuristics are described. We then discuss the development of Bayesian detector fusion and presentation and cataloguing of metadata. We finish with a performance evaluation and a summary.

A critical evaluation of our contributions and a discussion of promising directions for future research is given in Section 6.

2 Data Mining on the World Wide Web

2.1 Information Retrieval on the World Wide Web

The Web is a collection of hyperlinks, documents and other artifacts. The number of hyperlinks to a page can be used as a static measure of measure its popularity. Almost no structure is imposed on data contained in artifacts, and links between artifacts provide very little structure. Obtaining relevant information from the Web is therefore not always straightforward. The technical challenges associated with locating relevant information are significant, underlined by the fact that conferences such as TREC¹², and CIKM¹³ were started as long ago as 1992 and 1994, respectively, to study text retrieval and information management in large scale repositories. These studies are connected with text analysis, link analysis, cross language information retrieval, and more, and have spawned a variety of information services, from general-purpose and special-purpose search engines, to intelligent user specific agents that learn to find relevant information. In the following paragraphs, we touch on a few chosen studies; the references are far from comprehensive, and we suggest that the reader consult one of the many books available for a more thorough treatment of information retrieval. For a quick overview on what role context plays in information retrieval on the Web, and for an example of a preference-based user agent, we suggest [15, 16].

The lack of natural language understanding by computers is a major problem in information science. Even if machines were to understand natural language, the efficiency problems in processing natural language queries are prohibitive. Procuring information on the Web is even more difficult, since the Web is heterogeneous, dynamic, and the information sources are distributed. In addition, the amount of information available on the Web is vast (at last count at least 6TB), and the users' needs and abilities vary [2].

Traditional information repositories, such as libraries, require reverse cataloguing for efficient information retrieval. This reverse index, or catalogue, used in libraries facilitates finding *relevant*

¹²The Text REtrieval Conference can be found at <http://trec.nist.gov>

¹³The Conference on Information and Knowledge Management is at <http://www.cs.umbc.edu> or <http://cikm2001.cc.gatech.edu>

information. Two commonly used information measures that score the retrieval method over the catalogue are *precision* and *recall*; precision is the frequency of relevant artifacts retrieved, and recall is the frequency of relevant artifacts on total available artifacts. Applying these measures, and by looking at some statistics associated with the Web, the Web characteristics we expound on in the next paragraphs illuminates the magnitude of the problems faced by information retrieval technologies [2].

The heterogeneity of the Web can further be differentiated into a semi-structured aspect (for example the hyperlink structure, and document structure), a myriad of languages, and a wealth of artifact formats, ranging from the common HTML to obscure image, video and sound formats. The semi-structured link system of the Web has been studied with techniques from social networks by finding some weighting of the hyperlinks between web documents. Semi-structured data extraction techniques have been developed that utilise HTML structure, with some of the methods translating HTML into XML. Webseer, for example, is a niche search engine that provides an index of images based on text found with images, and image headers [17, 18, 19, 20, 21, 22].

The Web is a dynamic environment, with millions of pages being added, changed or removed every month. Alexa reported a doubling in the size of the Web every eight months, with 1.5 million pages added daily in 1998. Additionally, many web sites have dynamic content, which can cause an index of these sites to be permanently out of date, or simply not applicable. Alexa also reports that 90% of the Web traffic is to the top 100,000 web servers in 1998, and Lawrence and Giles reported that about 3 million web servers existed in 1999, pointing to the distributed nature of the Web. Some of the problems resulting from the characteristics of the Web include relevance, recall, and precision. We discuss in depth how search engines and web crawlers handle these aspects in Section 2.4. We also refer the interested reader to Florescu *et al.* for a survey on the database approach to information retrieval [2, 23, 24, 25].

We argue that special-purpose search engines are necessary in the Web as an information environment. Current general-purpose search engines cannot, for example, answer questions such as “Hand me a list of cameras that cost between one and two thousand Euros”. This limitation re-

sults from the methods these engines use to construct indices, and from the languages employed to query them. We believe that this type of limitation can be overcome by using more versatile query languages, and domain specific knowledge. The sections below touch on some of the technologies that have been proposed in the literature to overcome or at least alleviate many of the issues highlighted in this section.

2.2 Web Navigation

Internet users navigate the Web mostly by using a browser, but finding important pages is not always trivial, and a large amount of time can be wasted in following useless paths in the World Wide Web graph. In this section, we discuss some methods that are used to locate important pages first. We also note that an interesting variant on browsing is proposed by Lieberman *et al.*, where collaborative real time browsing is described, as a social exercise [26].

Kleinberg shows that “good” pages can be found by tapping into the structural information. A link to a document confers authority to that document. If A is the adjacency matrix of the Web’s structure, the principle eigenvectors of the matrices $A^T A$ and AA^T represent the authorities and hubs, respectively, where an authority is a document that is linked to by many good hubs, and a good hub points to many good authorities. The eigenvectors then provide a natural clustering [17].

Another way of ranking a document is explored with PageRank (see also Section 2.3), where the page rank PR of a web document i is given recursively by

$$PR(i) = dD(i) + (1 - d) \sum_{j \text{ linksto } i} PR(j)/N(j) \quad (1)$$

summed over all documents j linking to document i , $N(j)$ is the number of links contained in document j , D is a given probability distribution, and the estimate d takes a value between 0 and 1 [19, 27].

SALSA utilises random walks on graphs derived from the link structure to compute hubs and authorities; instead of employing the “mutual reinforcement” paradigm used by Kleinberg, a weighted

in-degree analysis approach is used. A bipartite graph consisting of authorities in the one part and the hubs in the other is assumed; a link from a hub to an authority forms an *information link*, and since these authority/hub documents are highly visible, they can then be found using random walks [28].

An in-depth comparison of these methods along with Bayesian models is done by Borodin *et al.* The authors show that both Kleinberg and SALSA have strengths and weaknesses, and propose a method for combining the two approaches. Specifically, Kleinberg's algorithm suffers from the Tightly Knit Community (TKC) effect, which is a side-effect of the mutual reinforcement approach used, resulting in topic drift in some cases. Intuitively, the topic drift is caused by the strong linking to authorities, and the authorities might not be authorities on the specific query, but are authorities on a related subject. SALSA ranks the authorities based on their popularity in the immediate neighbourhood, resulting in authorities that are not necessarily from the same community [29, 28].

2.3 Web Crawlers

Search engines need some efficient mechanism for document retrieval. This is usually done with an agent that traverses the hyperlinks found in web pages in some order, usually with agents running in parallel. The process of traversing and collecting web pages is called *crawling*, *trawling*, *spidering* or *robot scheduling*, and some of the older literature refers to the crawler as a *robot*, *walker*, *wanderer* or *worm*. Crawling the Web is a resource intensive activity, consuming significant bandwidth and computational resources. Methods for choosing important pages first have been investigated by, for example, Cho *et al.* One of the more important metrics is PageRank which defines the importance of a page to be the weighted sum of the back-links pointing to that page¹⁴. This metric is used by Google (Section 2.5). Web pages and the information therein change over time. Thus, search engines need to update their information by revisiting web sites. An analytical model for optimal scheduling of web crawlers has been developed [30, 31, 32].

The Web is increasing in size daily, and has recently been proven to surpass one billion docu-

¹⁴PageRank corresponds to the principal eigenvector of a normalised matrix of the Web (Sections 2.2, 2.5).

ments. The growth and size of the Web is thus a big limitation to small developers, and curbs the construction of new search engines. Before we discuss recent work in *focused* web crawlers aimed at alleviating this problem, see also Section 2.1, we note that at least one niche search engine¹⁵ chooses not to use a focused crawler; instead, it leverages existing general purpose search engines to obtain relevant documents. The reasons for this approach is that the categories used do not need a complete coverage and cost. [33].

Large-scale crawlers, used by big general purpose search engines, are mostly not publicly documented, since this could provide a competitive advantage in the commercial market. Interestingly, Google, one of the very successful search engines, is discussed in the literature. Google utilises a large-scale crawler, and the authors report that DNS (Domain Name Server) lookups are one of the big bottlenecks in crawling, since the crawling load is distributed among many personal computers. The *URLserver* passes lists of URLs (Uniform Resource Locator) to the distributed crawlers which then fetch and pass the documents to the *storeserver*. An ID number is assigned to every document, utilised by the indexer to create an inverted index [34].

2.4 Focused Web Crawlers

Traditionally, crawling on the Web is done by large search engines that attempt to exhaustively crawl the Web in order to index web documents, or to build a snapshot of the Internet. Some search engines claim to crawl the entire web in a few weeks. However, the rapid increase in the size of the Web is causing significant efficiency problems, and only a few of the biggest search engines can crawl the entire Web. Populating a specialised knowledge base using a breadth-first search wastes resources, and is out of reach of most small developers. Additionally, documents on the Web are heterogeneous in style and content and dynamic, constantly being changed, removed or added. *Focused crawlers* are a recent development created to alleviate these problems, and they do not require the significant resources used by the big search engines [35].

One of the problems faced by the general breadth-first crawlers is the dynamic properties of a large

¹⁵<http://www.researchindex.com>

component of the Web. The Internet Archive¹⁶ has archived 16 million Usenet postings from 1996-1998, which is about 600GB of (dynamic) data. This small component of the Web hints at the size of the dynamic content of the Web. The exponential growth in the size of the Web compounds this problem, and big search engines attempt to overcome this in two ways: resources and efficient crawlers [1, 36, 37].

Focused crawling was first suggest by De Bra *et al.* Implementation took a different form from contemporary crawlers: the crawler is implemented over Mosaic (the first web browser) in X (a Unix windowing system)¹⁷. The functionality provided in their Fish Search algorithm is regular expression matching, aimed at collecting only a small portion of the Web. For every document, every hyper-link in that document is a “fish”, and by following the link, the fish spawns new fish, where the new fish are the hyper-links of the new document. The hyper-link leads to a document that contains many hyper-links, or fish. There is a limit to the number of irrelevant documents that can be travelled before the fish dies of hunger [38].

WebWatcher uses learning methods to crawl the Web, and is the most cited system¹⁸ that uses focused crawling. A hyper-link is scored based on the TF-IDF (term frequency, inverse document frequency) score. WebWatcher applies machine learning techniques to choose which hyperlink to follow. A combination of four of their proposed methods for choosing a hyper-link yields the highest accuracy [16, 39].

2.4.1 ARACHNID: Adaptive Retrieval Agents Choosing Heuristic Neighbourhoods for Information Discovery

ARACHNID differs from the Fish Search algorithm, in that it uses a genetic algorithm based on *local selection*. Instead of comparing an agent’s fitness with other members of the population, some measure of fitness or *energy* is accumulated over time, and is compared with a fixed threshold. This

¹⁶<http://www.archive.org>.

¹⁷We note at this stage that focused crawlers are mostly used in personal agent systems, and in category specific search engines.

¹⁸From ResearchIndex.

property makes ARACHNID a distributed algorithm, since the decision to reproduce is local [40].

The algorithm is initialised by providing a list of keywords and a list of starting documents. The population of agents is then initialised by pre-fetching the starting documents, with each agent given a random behaviour, and an initial amount of “energy”. The relevance of the neighbouring documents is then assessed by analysing the text of the current document and a link is then followed based on the link relevance estimates. The agent receives energy, positive energy if the document is relevant, and negative energy for the cost of using the network. Agents also do not receive energy for documents already visited. Instantaneous changes of energy can be used as reinforcement signals, which allow an agent to adapt during its lifetime. Lastly, the agent can be killed or reproduce, and with reproduction, the agent is mutated depending on the method.

Two implementations were proposed using the above steps: a naive representation, and a vector representation. For the first representation, the two genotypes in \mathbb{R}^+ are initialised to some random values. A distance measure depending on the number of links between any two links, and keyword matching is used to compute a relevance score. A stochastic selector is then used to pick a link. Depending on the feedback of the user, the agent receives a positive or negative energy. The reward is weighted with a parameter indicating the trust assigned to the users’ judgement. Lastly, the offspring are computed based on the genotypes.

Each agent is represented by its genotype. This vector is initialised with the query terms over a vocabulary of words, and with a set of real-valued weights to a feed-forward neural network. One input is used for every query term, where the weights represent the relative importance of the query terms. The relevance of each link is estimated similarly to the naive implementation, but the agent’s keyword vector is used in place of the original query. The weighted frequencies of terms in the current document are presented to the neural network which computes the relevance estimate, and a stochastic selector is once again used to select an appropriate link. Energy assessments are obtained as in the naive method, but a list of encountered words is also maintained. The word counter is incremented or decremented based on the relevance score. The relevance score is computed using the hyperbolic tan on the frequency weighted sum of the TF-IDF weight of a term. The relevance

score is then used as a reinforcement signal for Q-learning, and the agent's network weights are updated. Finally the weights are mutated by adding random noise, and the word vector's terms are mutated with replacement.

Networked information environments are large, dynamic and not centralised, creating conditions where performing tasks is difficult. The author characterises this environment formally, and hypothesises that the Web has a *semantic topology*, where related documents are clustered by web authors use of hyper-links. The *generality* of a query is defined, as well as the *relevance autocorrelation*, and, using these definitions, shows that ARACHNID is well-behaved over a wide range of parameterisations. ARACHNID is compared with breadth-first search on the Encyclopedia Britannica. The system outperforms breadth-first search by a few orders of magnitude.

2.4.2 Using Reinforcement Learning to Spider the Web Efficiently

Reinforcement learning is a framework where decisions are learned by using rewards and punishments. The model is as follows: Let $s \in S$, S a set of states, $a \in A$, A a set of actions, $T : S \times A \rightarrow S$ a state-action transition function, and $R : S \times A \rightarrow \mathbb{R}$ a reward function. Maximising the sum of the reward for $\pi : S \rightarrow A$ is the *policy* we wish to learn. Let r_t be the reward received t time steps after state s by following policy π . A common formulation for reward over time is $V^\pi(s) = \sum_{t=0}^{\infty} \gamma^t r_t$ for $0 \leq \gamma < 1$, and the optimal policy maximises $V^\pi(s)$ for all states s . We learn the optimal value function V^* by learning the more specific correlate Q . We define $Q^*(s, a)$ to be the value of selecting action a from state s , and thereafter following the optimal policy, that is, $Q(s, a) = R(s, a) + \gamma V^*(T(s, a))$. The optimal policy in terms of Q can then be defined as $\pi^*(s) = \arg \max_a Q^*(s, a)$ [41].

Rennie and McCallum use reinforcement learning to create a topic-directed crawler by learning a mapping from text in the neighbourhood of a hyper-link to the expected number of relevant pages. The text is mapped to a scalar by casting regression as classification. A naive Bayes classifier then maps page content to one of a finite number of classes. A value is assigned to a hyper-link by weighing the average values of the highest ranked bins. The authors based their choice for

reinforcement learning on two factors. Performance is to be measured in terms of reward over time, and the World Wide Web as an environment presents a situation with delayed reward [42].

Learning the optimal policy for crawling directly is difficult, because of the size of the state space. The problem can be made tractable by assuming (1) the state is independent of the documents already visited, and (2) the relevant distinctions between actions is only dependent on the words in the “neighbourhood” of the hyper-link.

The Q -function is learned off line, and instead of using dynamic programming, a list of hyperlinks found in the sequence so far is maintained. The reward closest to this list is greedily traversed, and $\gamma \leq 0.5$ is chosen, since this value for γ results in an optimal expansion in the sense of traditional reinforcement learning.

A function that maps from hyperlinks to a scalar Q -value is chosen to be a collection of naive Bayes text classifiers. The classifiers are constructed as follows: Every class c_j has a document frequency $P(c_j)$ relative to all other classes. For every word w_t in the vocabulary V , the probability that w_t belongs to a given class c_j is written as $P(w_t|c_j)$. The naive Bayes assumption states that the words in the document d_i occur independently of each other, thus, the probability for each class given the document is $P(c_j|d_i) \propto P(c_j)P(d_i|c_j) \propto P(c_j) \prod_{k=1}^{|d_i|} P(w_{d_{ik}}|c_j)$. The word probabilities are estimated by computing the frequency with which w_t occurs in documents from class c_j . $P(c_j)$ and $P(w_t|c_j)$ are learned from a set of labelled training documents D . The estimate for the probability w_t in class c_j is

$$P(w_t|c_j) = \frac{1 + \sum_{d_i \in D} N(w_t, d_i)P(c_j|d_i)}{|V| + \sum_{s=1}^{|V|} \sum_{d_i \in D} N(w_s, d_i)P(c_j|d_i)}$$

where $N(w_t, d_i)$ is the number of times word w_t occurs in document d_i , and $P(c_j|d_i)$ is either 0 or 1, depending on the labelling. The class frequency is given by

$$P(c_j) = \frac{1 + \sum_{d_i \in D} P(c_j|d_i)}{|C| + |D|}$$

with $|C|$ the number of classes.

The experiments were performed on a fully mapped subsection of the Web. The authors experimented with a different number of classes. There is a tradeoff between classification accuracy and

the flexibility of the classifier-regressor; the three-bin crawler outperforms the two-bin crawler, and the four-bin crawler outperforms the three-bin crawler, however, the five-bin crawler performs worse – roughly equivalent to the two-bin crawler.

2.4.3 Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery

Chakrabarti *et al.* describe a focused crawler based on a canonical topic taxonomy, arguing that this approach results in a better modelling of the negative class, allowing reuse of classifier training effort, and discovery of related classes that are topically related, but not contained in the user's start set [43].

The focused crawler consists of a classifier, which does a document relevance test, a distiller that assigns document popularity or *prestige* based on the links between pages, and a crawler with dynamically reconfigurable priority controls governed by the classifier and distiller. The goal of the crawler is to visit as many possible relevant pages and as few irrelevant pages as possible. Thus, the average relevance is maximised.

For classification, a document is seen as a bag-of-words, and the category taxonomy induces a hierarchical partition on web documents. Let c be the class or the node we wish to assign document d to. Then

$$P(c|d) = P(c \cap \text{parent}(c)|d)$$

since we have from the tree structure that

$$(c \cap \text{parent}(c)) = c \Rightarrow (c \subseteq \text{parent}(c))$$

and subsequently

$$= \frac{P(c \cap \text{parent}(c) \cap d)}{P(d)} = \frac{P(\text{parent}(c) \cap d)}{P(d)} \frac{P(c \cap \text{parent}(c) \cap d)}{P(d \cap \text{parent}(c))}$$

. We then obtain

$$P(c|d) = P(\text{parent}(c)|d)P(c|d \cap \text{parent}(c))$$

To obtain $P(c|d \cap \text{parent}(c))$: The root node has by definition a probability $P(c_0|d) = 1$ that a document d is assigned to the root category c_0 . We can then write

$$P(c_i|d \cap c_0) = \frac{P(c_i \cap d \cap c_0)}{P(d \cap c_0)} = \frac{P(c_0)}{P(d \cap c_0)} \frac{P(c_i \cap c_0)}{P(c_0)} \frac{P(d \cap c_i \cap c_0)}{P(c_i \cap c_0)} = \frac{P(c_0)P(c_i|c_0)P(d|c_i \cap c_0)}{P(d \cap c_0)}$$

for a child category c_i .

The value for $\frac{P(d \cap c_0)}{P(c_0)}$ can be obtained by writing

$$\begin{aligned} \frac{P(d \cap c_0)}{P(c_0)} &= \frac{1}{P(c_0)} P(d \cap (\bigcup_k c_k)) = \frac{1}{P(c_0)} \sum_k P(d \cap c_k) = \frac{1}{P(c_0)} \sum_k P(c_0|c_k) P(d \cap c_k) \\ &= \frac{1}{P(c_0)} \sum_k \frac{P(c_k \cap c_0)}{P(c_k)} P(d \cap c_k) = \sum_k P(c_k|c_0) P(d|c_k) \end{aligned}$$

since the children c_k are disjunct, and $P(c_0|c_k) = 1$.

Since the document can be viewed as a bag-of-words, $P(d|c)$ can be estimated. The page generator uses $P(c|\text{parent}(c))$ to choose a leaf c^* . The probability $\theta(c^*, t)$ is the probability that term t turns up on a die, where the die has a face for every unique term in the universe. The generator chooses a length $n(d)$ for the document d , and flips the die repeatedly. If term t occurs $n(d, t)$ times then

$$P(d|c) = \binom{n(d)}{n(d, t)} \prod_{t \in d} \theta(c, t)^{n(d, t)}$$

The classifier is used to focus with the hard rule, where the leaf with the highest probability is found and if some ancestor of this leaf has been marked as good, future visitations of URLs found on the document d is allowed, otherwise the crawl is pruned at d . The soft focus rule guesses that the priority of visiting a neighbour is the relevance $R(d) = \sum_{\text{good}(c)} P(c|d)$ of the page d , where *good* indicates a page marked good, and with the assumption that a good node is never the ancestor of another.

The distiller is based on work done in social networks and has been adopted to hyperlink analysis (see Section 2.2 for details). Instead of assuming unary weights for the initial matrix, the forward and backward edge weights are differentiated into \mathbf{E}_F and \mathbf{E}_B , where the weight $\mathbf{E}[u, v]$ of edge (u, v) is the probability that u linked to v because v was relevant with relevance $R(u)$. The same is done for $\mathbf{E}_B[u, v]$ to prevent prestige incorrectly being assigned from an authority to a hub. Relevant authorities included in the graph are chosen depending on some threshold, typically between

10-20% of the most relevant nodes, and for hubs no such rule is used. We refer the interested reader to Kleinberg for detail, and note that the edge set is restricted to edges between different sites [17].

The performance of the crawler is evaluated with the *precision* benchmark from information theory. The rate of harvesting relevant pages is tested with the classifier developed above. The unfocused crawl results in very few relevant pages per URLs fetched, whereas the hard and soft focused crawl outperforms the unfocused crawl by orders of magnitude [43].

2.4.4 Focused Crawling Using Context Graphs

The links in web documents are uni-directional, and standard crawlers follow the resulting graph with some search method in the direction of the links. This search is called *forward crawling*. *Backcrawling*¹⁹ finds the links pointing to a given page. Diligenti *et al* use backward crawling in their Context Focused Crawler (CFC). The CFC models hierarchies with a so-called *context graphs*. The context graph generated from every seed document is then merged to form a *merged context graph* [14].

A context graph is generated by backcrawling a seed document. Every seed document is a layer 0 document. The pages found pointing to layer 0 are grouped into layer 1 in the context graph. Backcrawling from layer 1 results in layer 2, and so on, until a user-specified number of layers have been filled. Every document is a node connected with an edge to its parent. The backcrawling is done by querying search engines that provide the capability to find links pointing to the seed document, such as Google or AltaVista (Section 2.5). A statistical sample of parents' nodes is taken when the number of elements in a layer increases beyond some limit. Path strategies of up to N steps can be modelled with N levels in the context graph.

Naive Bayesian classifiers are trained for every layer of the context graph for quantifying the belief in assigning a document to that layer. The CFC uses TF-IDF (Term Frequency Inverse Document Frequency) on a vocabulary of phrases as the features. The TF-IDF score $\nu(w)$ of a phrase w is

¹⁹Backcrawling finds the documents that link to a specific web document, instead of following links found in that specific web document

given by $\nu(w) = \frac{f^d(w)}{f_{max}^d} \log \frac{N}{f(w)}$, where $f^d(w)$ is the number of occurrences of w in document d , f_{max}^d is the most common word in the document, N is the number of documents in the seed set and $f(w)$ is the number of documents where the phrase w occurs at least once.

To build the classifiers, the documents in the seed set (and optionally the first layer) are concatenated with the stop words (e.g. “of”, “a”, “an”, “the”, etc.) removed, word stemming performed and TF-IDF applied with the reference corpus derived from an exhaustive web crawl. The resulting set of phrases forms the vocabulary. The TF-IDF score for these phrases are computed and the forty highest ranking phrases are chosen.

A set of parallel two-class naive Bayes classifiers are built; one per layer, and the winning layer for a new document is selected by maximising the *a-posteriori* likelihood. $P(c_j)$ denotes the prior probability of documents from class c_j in layer j occurring on the Web. The probability of a phrase w_t in the reduced vocabulary occurring in documents of class c_j is $P(w_t|c_j)$. A page is assigned to class c_j for which $P(c_j|d_i)$ is maximised. The solution follows from Bayes rule

$$P(c_j|d_i) \propto P(c_j)P(d_i|c_j) \propto P(c_j)P(w_{d_i,1} \dots w_{d_i,N}|c_j)$$

where $w_{d_i,k}$ is the k -th feature of document d_i . Assuming features occur independently of each other, yields

$$P(c_j|d_i) \propto P(c_j) \prod_{k=1}^{N_{d_i}} P(w_{d_i,k}|c_j)$$

with N_{d_i} the number of features in d_i .

The maximum $P(c_j|d_i)$ over all the layers is then compared to a threshold value, and the winning document is kept or discarded. The documents in layer j of the context graph are combined to form the training set D_j . The phrase probabilities $P(w_t|c_j)$ are computed from the D_j sets by counting the number of occurrences of feature w_t and normalising for all the words in the documents of class c_j :

$$P(w_t|c_j) = \frac{1 + \sum_{d_i \in D_j} N(w_t, d_i)P(c_j|d_i)}{|V| + \sum_{d_i \in D_j} \sum_{s=1}^{|V|} N(w_s, d_i)P(c_j|d_i)}$$

where $N(w_t, d_i)$ is the number of occurrences of w_t in document d_i and $|V|$ is the number of

phrases in the vocabulary V . The parameters $P(c_j)$ are assumed to be the constant value $1/C$, where C is the number of layers.

The classifier of layer 0 is used to determine the topical relevance of a document, i.e. whether or not a page should be included in the result set. The remaining classifiers are used to predict the number of links to follow to find a target document. A queue is created for every layer and for the discarded documents. The queues are maintained in sorted, links contained in the first non-empty queue are followed.

The CFC is compared with a standard breadth-first crawler, and with a traditional focused crawler. In an experiment to find conference material, both the focused crawlers significantly outperformed the standard breadth-first crawler, but the CFC found on average 50-60% more on-topic documents than the standard focused crawlers. The chosen metric for “on-topic” is the fraction of relevant pages compared to the total number of download requests.

2.5 General Purpose Search Engines

The World Wide Web Worm was one of the first general purpose search engines which indexed approximately 110,000 documents on the Web in March 1994. The URLs, document names, and titles, as well as images were indexed [44].

Laser uses learning methods and HTML tags to improve the retrieval performance. Link information and a form of TF-IDF was used as parameters for a reinforcement learning scheme [45].

Sase investigated the use of a compressed index that represents a tradeoff between storage size and speed on a text database; it provides an exact and an approximate matching mechanism [46].

Lycos clusters query logs’ queries, and applies the clusters to new queries. Two queries are assumed equivalent if the same URL is followed. Two URLs are assumed equivalent if the same query is submitted. The clustering is then computed by iteratively grouping URLs and queries based on some similarity measure. Thus, equivalence classes are computed, yielding a set of relevant URLs for the queries [47].

Google utilises link information in order to improve the relevance of retrieved results, and to improve web coverage of its index. To improve the relevance of retrieved documents, a page receives a ranking based on link popularity (Section 2.3). The page is indexed on text associated with links pointing to that page, and with the text contained in the page. URLs are crawled by several distributed crawlers, where the URLs are obtained from the *URLserver*. The crawled pages are compressed and indexed using text and HTML structure, and the extracted URLs' metadata is stored. Google uses thousands of personal computers linked together to create a scalable search engine, differing from most of the other big general purpose search engines [19, 27, 34, 37].

2.6 Metasearch Engines

General-purpose search engines attempt to index a large part of the Web, and, despite heroic efforts in computing resources, still take a significant amount of time to crawl only a small portion. Differences in indexing algorithms and document crawl priorities influence the availability of newer and less popular documents. Search engines such as Google use the number of links pointing to a page as a popularity measure, thus, newer and less popular documents are less likely to be visited. The increase of the Web size, and the limits in hardware makes it almost impossible to index the entire web. Thus, general-purpose search engines are not able to index the entire web. *Coverage* is therefore lacking with general-purpose search engines. In 1999, Northern Light had the highest coverage at 16% of the estimated size of the Web (38.3% coverage with respect to combined coverage). The combined coverage of the search engines is 42%. To increase the coverage and currency of documents, users have to query more than one search engine. This search strategy results in 3.5 times as many documents to be found compared to using only one search engine. *Metasearch engines* can query multiple search engines, thus increasing currency and coverage, and provides a consistent user interface. However, combining information from multiple sources in a ranking order that makes sense is difficult. Source selection is another issue faced by metasearch engines, as is query modification [2, 48].

MetaCrawler is a prototypical metasearch engine. Experiments performed with MetaCrawler show

that single search services are not sufficient to the average user's need. Previous work on combining information from multiple sources is discussed by Arens *et al.* This knowledge base approach assumes domain knowledge, which can be a disadvantage [49, 50].

SavvySearch is a metasearch engine that increases coverage by querying specialised and general-purpose search engines. It penalises search engines by using their recent hits and response time. The meta-index used by SavvySearch is a vector that associates query terms and search engines. The effectiveness values in the vector is a modification of TF-IDF. The authors report an increase in queries submitted to appropriate search engines, and a decrease in querying inappropriate search engines [51, 52].

Gloss or *Glossary of Servers Server* provides a framework for studying metadata supplied by text sources. This metadata describes the text source, and when queried by the metasearch engine, provides a list of promising sources to query. The big advantage of this approach is the reduction in overhead, which offsets the potential drop in accuracy when a full query is supplied to every text source [53].

MetaSEEk is the first metasearch engine aimed specifically at retrieving information from six special-purpose search engines that index images. An architecture for an image metasearch is also proposed by Lawrence and Giles [54, 55].

Inquirus downloads documents in real time after querying general purpose search engines, and analyses these documents. A context summary around the query terms is displayed to the user, improving the efficiency of searching [56].

Inquirus-II is a new metasearch engine architecture that caters to users' information need in order to improve the precision and recall of the retrieved results. Both precision and recall improve with appropriate query modification. A query modification is a set of extra query terms added to a user query to improve the likelihood of relevant returned results. Returned results are scored based on the query terms and the user preference, and not scored on the modified query. Ranking the returned results is done by considering the full text of the documents, as opposed to traditional scoring that simply uses the ranking provided by the general purpose search engine. Query modifications for

each search engine is evaluated by using classifiers to improve the quality of the category specific or preference based search [13, 57].

A good metasearch engine has at least the following properties: it provides a consistent user interface, a comparatively high coverage, and a high recall and precision.

2.7 Special Purpose Search Engines

General-purpose search engines create indexes to increase the availability of web documents. Some of the problems associated with building such search engines were described in Section 2.5. Many of these can be solved with good software engineering and a big investment in hardware and bandwidth. However, they cannot improve coverage and currency. The one-size-fits-all approach does not solve all information needs. Most search engines allow keyword searches, but cannot handle queries such as “I need a list of cars that cost between 1000 and 2000 Euros, in and around Stellenbosch”. Special purpose – or niche – search engines that extract *metadata*, are becoming increasingly popular. Since the niche search engines index only a small portion of the Web, and exploit *domain knowledge*, these kinds of questions are more likely to be answered. Some of the issues that arise from this approach to web searching include choosing a correct information source for the query, obtaining a relevant set of objects with a high degree of precision, and extraction and indexing of data elements. The techniques for solving these problems automatically are important because of the savings in human time and effort. We first discuss search engines for computer science technical reports, CiteSeer²⁰ and Cora²¹. We then briefly mention search engines aimed at indexing multi-media content.

McCallum *et al.* investigate reinforcement learning, text classification, and information extraction techniques for automating the construction of a specialised search engine. The crawler used in this project is described in Section 2.4.2. Naive Bayes is extended to automatically assign documents to

²⁰<http://www.researchindex.com>

²¹<http://www.cora.whizbang.com>

a computer science hierarchy²². Hidden Markov models are used to extract metadata including the abstract, title, author, institution, and citations are extracted with an accuracy of 91-93%. We note that in subsequent work by Connan, the accuracy of extracting the bibliography has been increased significantly, to 97.7% and upwards in accuracy for the IEEE, AAI, and NEWAPA bibliography styles, and with more than 99% accuracy on individual fields, and with complete discrimination performance between these styles. Lawrence *et al* show that informal citations, such as hyperlinks, disappear quickly, and that formal references are preferable [58, 59, 60].

CiteSeer is a digital library which indexes computer science postscript and PDF technical reports. CiteSeer analyses citations using the hubs and authority approach for ranking (Section 2.2). The citations are extracted with various methods, including the *LikeIt* intelligent string comparison method, and word and phrase matching. Documents are compared for similarity by using the TF-IDF metric, the *LikeIt* string distance for headers of documents, and citations to other documents. The different metrics are then combined [61, 62, 63].

A different type of special-purpose search engine is aimed at indexing multi-media metadata. Image indexers range from pure image content based methods such as QBIC, methods that use both visual and textual cues such as WebSeer, to methods based only on text/HTML features, such as the work done by Lawrence and Giles [22, 64, 55].

VideoQ is an automated content based video search engine that exploits visual cues to retrieve visual clips, e.g. soccer players, skiers, and high jumpers. The system has the ability to automatically perform object segmentation and tracking, it contains a feature library, and allows spatio-temporal constraints to be specified. A content based video indexing system for TV news broadcasts is described by Eickeler & Müller, where the authors use hidden Markov models to recognise, for example, interviews, newscaster, and the weather forecast [65, 66].

We note that the special-purpose search engines differ widely in application and construction, though most are constructed with text or HTML source data in mind²³.

²²The principle is similar to Yahoo's hand coded hierarchy.

²³See e.g. <http://www.beaucoup.com> for a list of search engines.

2.8 Opportunities for Machine Learning

Machine learning investigates the mechanisms by which knowledge is acquired through experience. Some of the machine learning techniques used or described in this thesis includes Bayesian learning, naive Bayes (a modification of Bayes learning), support vector machines (SVMs), Q-learning, hidden Markov models and others.

Which properties of the Internet provides good opportunities for the application of machine learning techniques? We believe the dynamic, heterogeneous, and distributed nature of the Web is naturally suited for the application of machine learning techniques. Machine learning techniques are applied very successfully in the above sections, and opens opportunities for research into new areas. There is clearly a myriad of opportunities for machine learning in this information environment. Examples include web crawlers and search engines such as Google that uses the links between web documents to judge the importance of documents, and to decide which documents to crawl first. The freshness of documents can be learned, and re-crawling policies produced. Some metasearch engines learn which search engines to query based on a meta-index or some other term or word vector, and then estimating the relevance the documents in that text source has to the query. User preferences are also a common area for the application of learning, typically in the way users traverse the Web, query log analysis, and document similarity. Obtaining documents that are similar to some seed set is very useful in category specific search engines, and is used fruitfully in this thesis. The learned dynamics of document requests for caching could greatly improve caching schemes, and result in improved Internet bandwidth utilisation [13].

Machine learning techniques are therefore useful, but not necessarily the be- all and end-all, because many of the learning techniques do not explain why a certain decision is made (e.g. many neural network architectures). Other disadvantages can include computational and memory requirements and the fact that learning usually provides estimated results; the distribution underlying training data might, for example, be totally different from the distribution of the data the learning is applied to. Thus, the model is determined empirically, and not analytically. Data-preprocessing and prior knowledge can influence learning significantly and many applications of learning are

only really efficient after significant effort is spent on massaging data. Despite these and other factors, machine learning provides a cost-effective way for handling large amounts of data, and in many cases provides expert domain knowledge to many users that are potentially untrained in the domain in question. Machine learning techniques are tools, and if applied correctly, can yield a significant savings in human effort.

The strengths and weaknesses of learning techniques depend on the individual type of learning used, and is beyond the scope of this thesis. Learning techniques are increasingly used on the Web to facilitate data management and mining, as the following discussion in Section 2.9 shows.

2.9 Related Work

We compare DEADLINER with two other niche search engines that perform extensive field extraction, namely ResearchIndex and CORA [67, 58].

ResearchIndex, – also known as CiteSeer – and Cora are special-purpose search engines that index scientific literature on the Web. CORA uses Hidden Markov Models (HMMs) to extract information in citations. ResearchIndex, in contrast, uses hand-constructed algorithms and heuristics, where the most uniform features are first parsed and syntactic relationships are used to predict other fields. Both these systems required significant effort to construct, which could prove problematic when porting their architectures to new applications. DEADLINER has a simpler and we believe more flexible approach, which uses simple filters to select appropriate text blocks to narrow the text space and followed by the application of simple heuristics [62].

We note that traditional search engines do not index postscript and PDF documents, which both CiteSeer and Cora do. Further, Connan *et al.* reports on an improvement on the extraction results as compared to the HMMs used in CORA (see also Section 2.7)[59].

The DEADLINER architecture is not limited to metadata for text, since the Bayesian approach and the Neyman-Pearson Lemma are designed for testing a hypothesis H_0 against a hypothesis H_1 . Any set of filters that generate a labelling can be used. It is possible to create, for example,

an audio indexing system (Section 5). A few other approaches to extracting information is shortly described in Section 3.6.

2.10 Summary

The sheer size of the World Wide Web, with its large information space, result in poor coverage and currency. Special-purpose search engines circumvent many of the technical challenges associated with comprehensive search engines, and introduce some of their own. These challenges include focused crawling, and utilising domain knowledge. Machine learning is a viable approach to many of the technical challenges presented by focused crawlers, and for exploiting domain knowledge data mining on the web, coupled with information retrieval and search engines is a fascinating topic, and well worth studying in depth. The graph traversals the focused crawlers do, leads to novel graph algorithms, and interesting modifications of existing graph traversal algorithms. Examples of niche search engines abound, with some of the more well-known commercial engines include product comparison at MySimon²⁴, a dictionary, acronym, and thesaurus service²⁵, an MP3 search engine²⁶, and many more. These very successful niche search engines reveal a new direction for Internet research, that is, the automated construction of special-purpose search engines.

²⁴<http://www.mysimon.com>

²⁵<http://www.thesaurus.com> and <http://www.dictionary.com>

²⁶<http://www.mp3.com>

3 Document Classifiers

3.1 Introduction

Niche search engines, category specific engines, email-filters, and search engines with a topic hierarchy are just a few real-world examples where document classification can be fruitfully applied. We discuss a few of the frequently used document classification techniques. Comparison studies by Yang and Joachims indicate that the k -nearest neighbour technique (k -NN) performs well, but SVMs outperform any of the traditional techniques. Yang experimentally tested the suitability of 14 text classification methods. He found that the k -nearest neighbour (k -NN) algorithm (Section 3.4), a neural network architecture approach aimed at a specific corpus (Section 3.3), the Widrow-Hoff, and a linear least squares fit performs the best of the tested techniques. Work done by Joachims shows that SVMs outperform k -NNs, the naive Bayes approach, C4.5 and Rocchios Term Frequency to Inverse Document Frequency (TF-IDF) approach. We discuss the naive Bayes approach in Section 3.2, k -Nearest Neighbours (k -NN) in Section 3.4, and the neural network²⁷ approach on the Reuters collection in Section 3.3 [3, 68, 69].

Classification techniques aimed at classifying into a large number of categories include Kohonen's *self organising feature maps* (SOMs), and the work done by Koller and Sahami, where naive Bayes classifiers are hierarchically structured. SOM is an unsupervised clustering technique, and can be scaled to a large number of categories and features. The Bayesian hierarchy approach is organised such that every category becomes a simpler problem, and then easier to classify [70, 71].

3.2 Bayesian Learning

Bayesian learning is based on Bayes' Theorem, named after the English mathematician and theologian, the Reverend Thomas Bayes. The Reverend was born in 1702, died in 1761, and his famous pioneering work in probability was published post-humously in 1763. This work is set forth in an

²⁷Note that the neural networks approach is aimed at a specific corpus of documents, and was constructed with this corpus in mind.

“Essay Towards Solving a Problem in the Doctrine of Chances”, and forms the basis for many contemporary classifiers [72, 73].

Let $P(h)$ be the probability that hypothesis h holds before observing the data, also called the *prior probability*. Let $P(D)$ be the prior probability that training data D will be observed. Given some world in which h holds, $P(D|h)$ denotes the probability of observing data D if hypothesis h holds. Finally, let $P(h|D)$ be the probability that h holds given the observed data D , called the *posterior probability* of h .

Theorem 1 (Bayes) *The posterior probability $P(h|D)$ of h given D*

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Let $c_j \in C$ be a finite set of *classes*, and let every training instance x be described by a tuple $\langle w_1, w_2, \dots, w_n \rangle$ of attributes. The classification for each instance can be predicted from Bayes’ theorem. The computational complexity prohibits the use of a large number of attributes; however, if we make the *naive Bayes* assumption that the attribute values are conditionally independent given the target value, the probability of observing the events w_1, \dots, w_n is the product of the probabilities for the individual attributes $P(w_1, w_2, \dots, w_n|c_j) = \prod_i P(w_i|c_j)$. Using Bayes’ Theorem yields

$$P(c_j|w_1, w_2, \dots, w_n) = \frac{P(w_1, w_2, \dots, w_n|c_j)P(c_j)}{P(w_1, w_2, \dots, w_n)} = \frac{P(c_j) \prod_i P(w_i|c_j)}{P(w_1, w_2, \dots, w_n)}$$

To classify a document, we choose the class with the maximum posterior probability, and we note that $P(w_1, w_2, \dots, w_n)$ is a constant, yielding $c_{MAX} = \operatorname{argmax}_{c_j \in C} P(c_j) \prod_i P(a_i|c_j)$.

The probability of word w_i in class c_j can be estimated with the Laplacean²⁸ prior, avoiding “veto” classifications as a result of the product calculation.

The model we describe is a multi-variate Bernoulli model, with no dependencies between terms. A different model uses term counts to improve classification. In the experiments done by McCallum

²⁸Probability estimations are primed with a count of one.

et al., the multinomial model performs significantly better than the multi-variate Bernoulli model on larger vocabularies. We believe the reasons for the widespread use of the naive Bayes classifiers is its implementation simplicity and computational efficiency, since many other outperform these classifiers in accuracy [74, 68, 3].

3.3 Neural Networks

Neural networks can be applied successfully to document classification. Wiener *et al.* approaches feature selection for dimension reduction by choosing a subset of the original document terms – also referred to as Latent Semantic Indexing (LSI) – which constructs new features by combining a number of the original terms. Every vector term is computed

$$p_j = \frac{\sqrt{f_j}}{\sqrt{\sum_{vector} (\sqrt{f_i})^2}}$$

where f_i and f_j the word frequency. For feature selection, a *relevancy score* $r_k = \frac{w_{tk}/d_t + 1/6}{w_{tk}/d_t + 1/6}$ is computed with w_t the number of documents containing the term, and d_t the total number of documents with the topic. Terms that yield a high negative or positive value provide good classification. Singular value decomposition (SVD) is applied to the term by document matrix in order to perform LSI. To improve the performance of LSI, the documents are grouped into metatopics, and SVD is applied. The rational behind this strategy is to give more importance to lesser used terms. Standard back-propagation is applied to two different architectures. The flat architecture is a small (six units in the hidden layer) neural network, with a network trained for every topic. The modular architecture is designed such that the problem is divided into a set of smaller tasks. The first component tests whether or not a metatopic is present. The second component is a set of five network groups that each corresponds to a metatopic [75].

3.4 k -Nearest Neighbour

The k -nearest neighbour algorithm can be applied to both real and discrete valued target functions. We consider the discrete case for text classification. k -nearest neighbour algorithm finds the k exemplars closest to the new instance x that needs to be classified. The classifier function is $f : \mathbb{R}^n \rightarrow C$, with C the finite set $\{c_1, \dots, c_n\}$ of document classes. We define $\delta(a, b) = 1$ if $a = b$, and $\delta(a, b) = 0$ if $a \neq b$. The target function f is then estimated with \bar{f} by choosing the class that occurs most

$$\bar{f} \leftarrow \operatorname{argmax}_{c \in C} \sum_{i=1}^k \delta(c, f(x_i))$$

in those k closest exemplars.

The metric used with exemplars with numeric attributes is typically Euclidian. TF-IDF (Term Frequency - Inverse Document Frequency) is used many times for text classification. TF on a term w_i is the number of times w_i occurs in document d . IDF can be written as $\log \frac{D}{DF(w_i)}$ with D the number of documents and $DF(w_i)$ the number of documents w_i occurs in at least once. A document is then represented as a vector [41, 76].

3.5 Support Vector Machines

3.5.1 Separable Patterns and Building a Support Vector Machine

One of the problems with text classification is the size of the feature space. Dimension reduction is usually done before classification. Support vector machines (SVMs) integrate classification and dimension reduction, and are aimed at binary classification. SVMs fit a hyperplane as the decision surface so that the margin of separation between the positive and negative training samples is maximized. Formally, let $T = \{(\mathbf{x}_1, o_1), \dots, (\mathbf{x}_n, o_n)\}$ be the set of training examples with \mathbf{x}_i the input vector and $o_i \in \{+1, -1\}$ the desired output. For linearly separable classes, the decision surface in the form of a hyperplane for input vector \mathbf{x} , weight vector \mathbf{w} , and bias b is given by $\mathbf{w}^T \mathbf{x} + b = 0$. We then have $\mathbf{w}^T \mathbf{x}_i + b \geq 0$ for $d_i = 1$ and $\mathbf{w}^T \mathbf{x}_i + b < 0$ for $d_i = -1$. One

difference between SVMs and other classifiers is that the hyperplane is computed for the closest data point, which is the most difficult to classify. Let ρ be the *margin of separation*, i.e., the distance between the hyperplane and the closest data point. The hyperplane that maximises the margin of separation provides the best classification performance, and is called the *optimal hyperplane*. This hyperplane in the input space is given by $\mathbf{w}_o^T \mathbf{x} + b_o = 0$. The function $f(\mathbf{x}) = \mathbf{w}_o^T \mathbf{x} + b_o$ is a measure of the distance from \mathbf{x} to the optimal hyperplane. Rewrite \mathbf{x} as the normal projection \mathbf{x}_p onto the optimal hyperplane, with r the desired distance: $\mathbf{x} = \mathbf{x}_p + r \frac{\mathbf{w}_o}{\|\mathbf{w}_o\|}$. Substituting into $f(\mathbf{x})$ yields $f(\mathbf{x}) = \mathbf{w}_o^T \mathbf{x} + b_o = r \|\mathbf{w}_o\| + \mathbf{w}_o^T \mathbf{x}_p + b_o = r \|\mathbf{w}_o\|$ since by definition $f(\mathbf{x}_p) = 0$ [77, 78, 3].

We wish to compute the optimal hyperplane's parameters \mathbf{w}_o and b_o for the training set T . Thus, $\mathbf{w}_o^T \mathbf{x}_i + b_o \geq 1$ for $d_i = 1$, and $\mathbf{w}_o^T \mathbf{x}_i + b_o \leq -1$ for $d_i = -1$, with $1 \leq i \leq n$. The *support vectors* are the data points (\mathbf{x}_i, d_i) for which equality holds in these two equations. Since $r = \frac{f(\mathbf{x})}{\|\mathbf{w}_o\|}$, we have for a support vector \mathbf{x}_s for which $d_s = 1$ that $f(\mathbf{x}_s) = \mathbf{w}_o^T \mathbf{x}_s + b_o = 1$ yields $r = \frac{1}{\|\mathbf{w}_o\|}$. A support vector for which the desired output is -1 , is handled similarly. The optimum value for the margin of separation p is $p = 2r$, thus $p = \frac{2}{\|\mathbf{w}_o\|}$. The optimal separation between the positive and negative examples minimises the norm. The optimal hyperplane can be solved using quadratic optimisation [77, 78].

To find the optimal hyperplane in parameters \mathbf{w}^{29} and b , and given the training sample $T = \{(\mathbf{x}_1, o_1), \dots, (\mathbf{x}_n, o_n)\}$, constrained by $d_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$ for $i = 1, \dots, N$, we need to minimise the cost function $\psi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$. This problem has *linear constraints* and $\psi(\mathbf{w})$ is a convex function, and is therefore a *primal problem*. This problem may be solved with the method of *Lagrange multipliers* by constructing the Lagrangian function $L(\mathbf{w}, b, \alpha) = \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^N \alpha_i [d_i(\mathbf{w}^T \mathbf{x}_i + b) - 1]$. The variables α_i are called *Lagrange multipliers*. The saddle point of L gives the optimal solution. The conditions for optimality are then $\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = \mathbf{0}$ and $\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial b} = 0$. The first condition yields $\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = \mathbf{0} = (2)(\frac{1}{2})\mathbf{w} - \sum_{i=1}^N \alpha_i d_i \mathbf{x}_i \implies \mathbf{w} = \sum_{i=1}^N \alpha_i d_i \mathbf{x}_i$. The second condition yields $\sum_{i=1}^N \alpha_i d_i = 0$. The function $L(\mathbf{w}, b, \alpha)$ is convex, which means that the solution vector \mathbf{w} is unique. We note that only the α_i 's subject to the constraint $\alpha_i [d_i(\mathbf{w}^T \mathbf{w} + b) - 1] = 0$ for $i = 1, \dots, N$ that occurs at the saddle point can assume nonzero values, since the constraints

²⁹Note that \mathbf{w} is a transformation of \mathbf{w}_o , and the two matrices are not necessarily equal.

vanish at the saddle point [77, 78].

The duality theorem states that if the primal problem has an optimal solution then the dual problem has an optimal solution, and the corresponding optimal values are equal. For \mathbf{w}_o to be an optimal primal solution, and α_o an optimal dual solution, it is necessary and sufficient that \mathbf{w}_o is feasible for the primal problem, and $\psi(\mathbf{w}_o) = L(\mathbf{w}_o, b_o, \alpha_o) = \min_{\mathbf{w}} L(\mathbf{w}, b_o, \alpha_o)$. In order to state the dual, expand $L(\mathbf{w}, b, \alpha)$'s terms. We then have $L(\mathbf{w}, b, \alpha) = \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^N \alpha_i d_i \mathbf{w}^T \mathbf{x}_i - b \sum_{i=1}^N \alpha_i d_i + \sum_{i=1}^N \alpha_i$. The term $b \sum_{i=1}^N \alpha_i d_i$ is zero, because of the optimality condition. Also, $\mathbf{w}^T \mathbf{w} = \sum_{i=1}^N \alpha_i d_i \mathbf{w}^T \mathbf{x}_i = \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j$. The objective function Q can then be written $L(\mathbf{w}, b, \alpha) = Q(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j$ [77, 78].

The dual problem is finding the Lagrange multipliers $\{\alpha_1, \dots, \alpha_N\}$ that maximise the function $Q(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j$, given training data $T = \{(\mathbf{x}_1, d_1), \dots, (\mathbf{x}_N, d_N)\}$, and the constraints $\sum_{i=1}^N \alpha_i d_i = 0$ and $\alpha_i \geq 0$ for $i = 1, \dots, N$ [77, 78].

Finally, once we obtained the optimal Lagrange multipliers $\alpha_{o,i}$, the optimum weight vector \mathbf{w}_o can be computed with $\mathbf{w}_o = \sum_{i=1}^N \alpha_{o,i} d_i \mathbf{x}_i$. The optimum bias b_o can be computed with $b_o = 1 - \mathbf{w}_o^T \mathbf{x}_s$ for $d_s = 1$ [77, 78].

The generalisability of support vector machines is related to the complexity of the optimal hyperplane. The *Vapnik-Chervonenkis dimension* (VC dimension) is a measure of this complexity, and is defined on an ensemble of dichotomies³⁰ F to be the cardinality of the largest set that is shattered³¹ by F . To apply the method of structural risk minimisation, the hyperplanes' training classification error and the VC dimension should both be minimised. Vapnik proved that the set of optimal hyperplanes described by the equation $\mathbf{w}_o^T \mathbf{x} + b_o = 0$ has a VC dimension h bounded by $h \leq \{\lceil \frac{D^2}{\rho^2} \rceil, m_o\} + 1$, with D the diameter of the smallest ball containing all the input vectors, the margin of separation $\rho = \frac{2}{\|\mathbf{w}_o\|}$, and the dimensionality of the input space m_o . This theorem shows that, with a proper choice for the margin of separation ρ , the VC dimension (i.e. the complexity) can be controlled [79, 77].

³⁰A *dichotomy* is a binary classification function or rule that partitions a set of input vectors $V = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ into two disjoint subsets.

³¹A set of input vectors V is *shattered* if all possible dichotomies of V can be induced by functions in F .

3.5.2 Non-separable Patterns and Support Vector Machines

The case for optimal hyperplanes non-separable patterns is more complex; we give an outline for this case. The margin of separation between classes is *soft* if a data point violates the condition $d_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$ for $i = 1, \dots, N$. This violation can result in either correct classification, if the data point is on the right side of the decision surface, or in misclassification otherwise. *Slack variables* $\zeta_i, i = 1, \dots, N$ measure the deviation of a data point from ideal pattern separation:

$$d_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \zeta_i, i = 1, \dots, N \quad (2)$$

Clearly, if $0 \leq \zeta_i \leq 1$, the data point lies on the correct side of the decision surface, and for $\zeta_i \geq 1$ it lies on the wrong side. The support vectors are those vectors that satisfies this equation exactly³².

Minimising

$$\Phi(\zeta) = \sum_{i=1}^N I(\zeta_i - 1)$$

where

$$I(\zeta) = \begin{cases} 0, & \zeta \leq 0 \\ 1, & \zeta > 0 \end{cases}$$

with respect to weight vector \mathbf{w} and Equation 2, yields a separating hyperplane for which the misclassification error averaged over the training set is minimised. Computationally this minimisation is intractable, and we therefore approximate $\Phi(\zeta)$ with $\Phi(\zeta) = \sum_{i=1}^N \zeta_i$. The computation can be simplified by writing $\Phi(\mathbf{w}, \zeta) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^N \zeta_i$. The parameter C controls the tradeoff between the complexity of the machine and the number of non-separable points (i.e. the potential to overfit). The primal problem is then: Given the training data T , find the optimum values of \mathbf{w} and b such that $\Phi(\mathbf{w}, \zeta) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^N \zeta_i$ is minimised in the variables \mathbf{w} and ζ_i , and \mathbf{w} and b is subject to the constraints $d_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \zeta_i$ for all i and $\zeta_i \geq 0$ for all i . A derivation similar to the one above yields the dual problem: Given training data T , find the Lagrange multipliers that maximise the function $Q(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j$ constrained by $\sum_{i=1}^N \alpha_i d_i = 0$ and $0 \leq \alpha_i \leq C$ for $i = 1, \dots, N$, with C a user-specified parameter. Similar to

³²The decision surface would change if ζ_i is not included in the training set.

the derivation above, the optimum weight vector is given by $\mathbf{w}_o = \sum_{i=1}^{N_s} \alpha_{o,i} d_i \mathbf{x}_i$, where N_s is the number of support vectors. Again, at the saddle point we have

$$\alpha_i [d_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 + \zeta_i] = 0 \quad (3)$$

for $i = 1, \dots, N$ and $\mu_i \zeta_i = 0$ for $i = 1, \dots, N$, where the μ_i are the Lagrange multipliers that force non-negativity of the slack variables ζ_i . Since the derivative of the Lagrangian function for the primal problem with respect to ζ_i is zero, we have $\alpha_i + \mu_i = C$, and combined with the previous equation we have $\zeta_i = 0$ if $\alpha_i < C$. The optimum bias b_o can be computed from any data point substituted into Equation 3.

3.5.3 Building a Support Vector Machine

Cover's theorem states that a multidimensional (in our case the input space) space may be transformed into a new feature space where the training data are linearly separable with high probability. The conditions for this theorem is a non-linear transformation, and a high dimensional feature space. To build an SVM, we map the input vector into a high-dimensional feature space, and then construct an optimal separating hyperplane. This hyperplane is built in the feature space, and not in the original input space, since in this way we can minimise the VC dimension and provide generalisation. In the following paragraphs we provide detail on how to do this.

In order to extend the results in the previous section, we generalise to an *inner-product kernel* and we apply Mercer's Theorem to characterise appropriate kernels.

Let \mathbf{x} be a vector from the input space with dimension m_0 , and let $F = \{\phi_1(\mathbf{x}), \dots, \phi_{m_1}(\mathbf{x})\}$ be a set of nonlinear transformations into the feature space, with m_1 the dimension of the feature space. The decision surface in the form of a hyperplane $\sum_{i=1}^{m_1} w_i \phi_j(\mathbf{x}) + b = 0$ with a set of linear weights $\{w_1, \dots, w_{m_1}\}$ and bias b , can be written as $\sum_{i=0}^{m_1} w_i \phi_j(\mathbf{x}) = 0$, with $\phi_0(\mathbf{x}) = 1$ for all \mathbf{x} . The bias b is then written as w_0 . If we then define $\phi(\mathbf{x}) = [\phi_0(\mathbf{x}), \phi_1(\mathbf{x}), \dots, \phi_{m_1}(\mathbf{x})]^T$ we can write the image from the input space induced into the feature space as $\mathbf{w}^T \phi(\mathbf{x}) = 0$. Similar to the simplified derivation, we have from the optimal hyperplane with the partial derivative over \mathbf{w} of the

Lagrangian function equal to zero that $\mathbf{w} = \sum_{i=1}^N \alpha_i d_i \phi(\mathbf{x}_i)$. Substituting into the image equation, we have the decision surface in the feature space as $\sum_{i=1}^N \alpha_i d_i \phi^T(\mathbf{x}_i) \phi(\mathbf{x}) = 0$. The *inner-product kernel* is now defined as $K(\mathbf{x}, \mathbf{x}_i) = \phi^T(\mathbf{x}) \phi(\mathbf{x}_i) = \sum_{j=0}^{m_1} \phi_j(\mathbf{x}) \phi_j(\mathbf{x}_i)$ for $i = 1, \dots, N$ ³³. The decision surface, written with the kernel function then becomes $\sum_{i=1}^N \alpha_i d_i K(\mathbf{x}, \mathbf{x}_i) = 0$.

Mercer's theorem provides a way for identifying suitable kernel functions: Let $K(\mathbf{x}, \mathbf{x}')$ be a continuous symmetric kernel that is defined in the closed interval $\mathbf{a} \leq \mathbf{x} \leq \mathbf{b}$ and likewise for \mathbf{x}' . The kernel $K(\mathbf{x}, \mathbf{x}')$ can be expanded in the series $K(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}) \phi_i(\mathbf{x}')$ with positive coefficients, $\lambda_i > 0$ for all i . For this expansion to be valid and for it to converge absolutely and uniformly, it is necessary and sufficient that the condition $\int_{\mathbf{b}}^{\mathbf{a}} \int_{\mathbf{b}}^{\mathbf{a}} K(\mathbf{x}, \mathbf{x}') \Psi(\mathbf{x}) \Psi(\mathbf{x}') d\mathbf{x} d\mathbf{x}' \geq 0$ holds for all $\Psi(\cdot)$ for which $\int_{\mathbf{b}}^{\mathbf{a}} \Psi^2(\mathbf{x}) d\mathbf{x} < \infty$. We note that the dimensionality of the feature space can be infinitely large, and that the image of the decision surface in the feature space is linear, even though the input space is of finite dimension, and can be non-linear. The dual problem can now be stated, and instead of using the inner product $\mathbf{x}_i^T \mathbf{x}_j$, we use the kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$. Given the training sample T , find the Lagrangian multipliers $\{\alpha_1, \dots, \alpha_N\}$ that maximise the objective function $Q(\alpha) = \sum_{i=1}^N \alpha - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j K(\mathbf{x}_i, \mathbf{x}_j)$ subject to the constraints $\sum_{i=1}^N \alpha_i d_i = 0$ and $0 \leq \alpha_i \leq C$ for $i = 1, \dots, N$, and C a user-specified positive parameter. The optimum values of the Lagrange multipliers $\{\alpha_{o,1}, \alpha_{o,N}\}$ can then be used to find the optimum values of the weight vector \mathbf{w}_o with $\mathbf{w}_o = \sum_{i=1}^N \alpha_{o,i} d_i \phi(\mathbf{x}_i)$ ³⁴.

Examples of support vector machines includes the polynomial learning machine $(\mathbf{x}^T \mathbf{x}_i + 1)^p$, the radial basis function network $\exp(-\frac{1}{2\sigma^2} \|\mathbf{x} - \mathbf{x}_i\|^2)$, and the two-layer perceptron $\tanh(\beta_0 \mathbf{x}^T \mathbf{x}_i + \beta_i)$ ³⁵.

Latent Semantic Indexing is a technique used by Kwok with an SVM to improve accuracy, and consistently outperforms an SVM that does not use LSI. Synonymy and polysemy are problems that underlie document classification. LSI would, rather than using terms, represent documents using the latent concepts referred to by terms. These hidden structures can be found with singular value

³³Note that the kernel is symmetric in its parameters.

³⁴Note that the first component of \mathbf{w}_o denotes the optimum bias b_o .

³⁵The two-layer perceptron kernel is only valid for certain values of β_0 and β_1 .

decomposition of the term-document matrix A . The low-dimensional space obtained is spanned by the few eigenvectors of $A^T A$ that correspond to the largest eigenvalues, leading to the most important correlations between terms. This technique then also ensures dimension reduction. In Section 5, we use a different technique to achieve dimension reduction, and thus improve efficiency, where an entropy measure determines the most relevant terms [78, 80].

3.6 Information Extraction

In our development of extractors using the DEADLINER framework, most of the filters were simple regular expressions. The option exists to use the wealth of existing approaches for learning regular expressions or HMMs automatically [81, 82, 83]. However, we believe that labelling of sufficient data at the target field level for these approaches to work is more effort than having designers develop these expressions, and using a more modest data set for performing the integration. Some of the other approaches are mentioned in the following paragraphs.

Finite state transducers (FSTs) are used by Hsu for token extraction. A heuristic is applied to prevent non-determinism in the FST, and contextual rules are produced by an induction algorithm. The FSTs obtained are applied to HTML pages³⁶ for data extraction [84].

A wrapper construction system for transforming HTML pages into XML is XWrap. Rules are generated and applied to HTML, and interesting document regions are identified via an interactive interface. The same is done for semantic tokens, followed by a hierarchy determination for the content, resulting in a context free grammar. One of the goals of this system is minimal user interaction [85].

Stalker is an algorithm that uses landmark automata to generate wrappers. It is a greedy sequential covering algorithm, and tries to form a landmark automaton that accepts only true positives by iterating until it finds a perfect disjunct or runs out of training examples, where the best disjunct is the one that covers the most positive examples. New disjuncts are added iteratively to cover

³⁶DEADLINER, in contrast, was tested on straight text.

uncovered positive candidates [86].

3.7 A Case for a Hybrid Approach

We argue that a modular, hybrid approach to document classification and extraction is preferable when building a special-purpose search engine. Smaller problems are in most cases easier to solve than the complex bigger ones. The widely known Towers of Hanoi problem is such an example, where the object of the puzzle is to move N stacked disks from one of the three pegs to one of the other pegs. The disks, all of different radii, are stacked in strictly decreasing size. A bigger disk may never fit on a smaller disk. The divide-and-conquer strategy used to solve this problem, is widely used in computer science, and is an accepted method for solving many diverse problems. This approach frequently leads to efficient and flexible solutions that can be generalised. Further, all the other documented special-purpose search engines we are aware of separate these functions, and, lastly, we believe the maxim “Do one thing, and do it well” applies to solving problems that can be divided into smaller problems [61, 58].

We note that single, monolithic approaches have been attempted successfully, but to change to a new domain significant portions of the system needs to be modified. In our system (Section 5), only the information extraction component need to be altered. Added to this advantage, the training data for document classification need not be done by a computer scientist or a linguistics expert, since the documents need a simple binary labelling [87].

SVMs do very well at the classification task, and we chose this method for document classification. For information extraction we use Bayesian integration of feature detectors. Implementation details can be found in Section 5.

4 ROCs and Detection/Decision Theory

4.1 Introduction

This section aims to provide an introduction to the detection/decision theory and resulting Receiver Operating Characteristics (ROCs) theory used in Section 5. Decision/detection theory is used in very disparate academic fields, including psychology, medical diagnosis, computer science, and the traditional signal detection fields (e.g. optics, radar, sonar, etc.) for the detection of a signal in a noisy medium. We do not discuss the proofs of the well-known theorems presented here, but rather concentrate on the application of the theorems to yield usable mechanisms for the detection of signals amidst noise. We first present the general idea behind signal detection, and then apply this theory to construct ROC curves. ROC curves provide an efficient and accurate method for detecting signals, and are used in Section 5 to extract information from web documents [88, 89, 12].

To detect a signal in a specific trial, we divide the *observation interval* into two categories of intervals: an interval containing a signal and noise, and an interval containing noise alone. The detection process of the presence of a signal involves an event, evidence, and a decision. The event is present or is not present. The evidence depends on the probability distributions of the noise alone and of the signal added to the noise. Following observation, a decision of “yes” if the signal was present, or “no” if the signal was not present, must be made. This detection problem where only two events and two responses are possible, forms the basis on which ROC curves are built. We name the four important event-response conjunctions: (*signal*, “yes”) called a hit (correct acceptance), (*signal*, “no”) called a miss (incorrect rejection), (*noise*, “no”) named correct rejection, and (*noise*, “yes”) which is incorrect acceptance. For the purpose of this discussion, we assume that trials are independent of one another. We define the *a priori* probabilities P_s and P_n as the relative frequency of signal and noise, and relative frequency of noise alone, respectively. The conditional probability $P(Y|s)$ is the hit rate, or the probability that “yes” has been chosen, given that the signal s occurred in noise. Let n indicate noise with no signal. Similarly the false alarm rate $P(Y|n)$, and the probabilities $P(N|n)$ and $P(N|s)$ are defined. The likelihood ratio

$L(x) \equiv P(x|s)/P(x|n)$ summarises the change in the ratio dependent on the discrete distributions of the probability densities. The likelihood ratio measures the strength of the evidence x in a given observation, and is independent of the *a priori* probabilities $P(s)/P(n)$:

$$\begin{aligned} P(s \cap x) &= P(x|s)P(s) = P(s|x)P(x) \\ \Rightarrow \\ P(s|x) &= \frac{P(x|s)P(s)}{P(x)} \\ &= \frac{P(x|s)P(s)}{P(x|s)P(s) + P(x|n)P(n)} \end{aligned}$$

Invert, and divide by $P(x|s)P(s)$:

$$P(s|x) = \left(1 + \frac{1}{L(x)} \frac{P(n)}{P(s)} \right)^{-1} \quad (4)$$

This equation also shows that the posterior probability $P(s|x)$ is strictly monotone with $L(x)$. Note that the domain of $P(s|x)$ is 0 to 1, the domain of $L(x)$ is 0 to ∞ , and that $L(x)$ is not a probability.

4.2 Bayes Criterion

Every event-response conjunction has an associated cost. For example, acting on a signal may cost more than ignoring the signal. We define the expected value function:

$$\mathcal{E} \equiv P(Y|s)P(s)V_{s,Y} + P(Y|n)P(n)V_{n,Y} + P(N|n)P(n)V_{n,N} + P(N|s)P(s)V_{s,N} \quad (5)$$

where the V -containing variables is the cost associated with choosing a specific event-response conjunction. To maximise the expected value, the observer must choose the response that adds on average the most to the expected value for each trial.

Let x be a specific value of a random variable X . If the observer knows the prior probability $P(s)$, and the likelihood ratio $L(x)$, then from Equation 4, the observer knows the posterior probability $P(s|x)$. The expected value depends on the decision the observer makes. The expected value, given x and Y is then $\mathcal{E}(V|x, Y) \equiv P(s|x)V_{s,Y} + P(n|x)V_{n,Y}$ for the “yes” decision, and for the “no”

decision we have $\mathcal{E}(V|x, N) \equiv P(n|x)V_{n,N} + P(s|x)V_{s,N}$. The observer should choose “yes” if $\mathcal{E}(V|x, Y) > \mathcal{E}(V|x, N)$. That is, if

$$\frac{P(s|x)}{P(n|x)} > \frac{V_{n,N} - V_{n,Y}}{V_{s,Y} - V_{s,N}} \quad (6)$$

then choose “yes”, otherwise choose “no”.

Rewriting the decision rule of Equation 6 in terms of likelihood ratio and using *a priori* probabilities, we choose “yes” if

$$L(x) > \frac{P(n)}{P(s)} \frac{V_{n,N} - V_{n,Y}}{V_{s,Y} - V_{s,N}} = \eta \quad (7)$$

and otherwise we choose “no”. A decision rule based on likelihood ratio is called a *likelihood criterion*. Note that in case of equality in Equation 7, it does not matter which response is chosen; we have assigned it to the rejection set. The critical value η (cutoff or threshold) results in optimal performance with respect to the decision goal. Another point to remember is that, regardless of the dimensionality of x , the likelihood ratio gives a one-dimensional decision value.

To maximise the percentage of correct decisions (where the correct decisions are (*signal, yes*) and (*noise, no*); incorrect decisions are any other decisions), we use the posterior probabilities. Note that if $P(s|x) > .5$, then it is more likely for s to have occurred than n . That is, if

$$\frac{P(s|x)}{P(n|x)} = \frac{P(s)}{P(n)} L(x) > 1$$

then choose yes, otherwise no. Rewriting, we have “yes” if

$$L(x) > \frac{P(n)}{P(s)}$$

and otherwise choose “no”. An example of the performance based on different curves is shown in Figure 4.2.

4.3 Neyman-Pearson

To by-pass the use of *a priori* probabilities and costs, we can use *conditional probabilities*. The Neyman-Pearson observer maximises the hit rate ($P_d = P(Y|s)$) for a given false-alarm rate

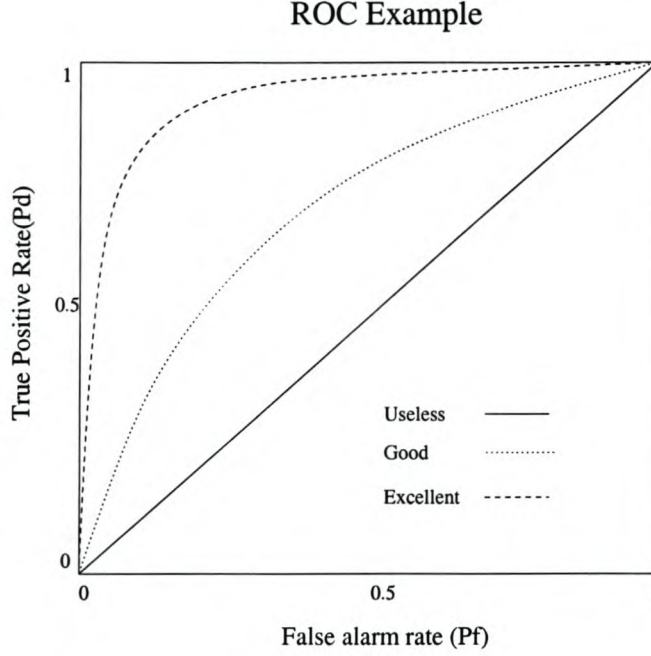


Figure 1: An example of a ROC. The straight chance line shows where $P(Y|s) = P(Y|n)$, and is the worst possible performance. The line starts at $(0, 0)$ (the empty set), and finishes at $(1, 1)$ (the full set). The ROC lies above the $P_d = P_f$ line.

$(P_f = P(N|s))$, under the constraint that the false-alarm rate does not exceed a fixed value α . The decision rule can be shown to be a likelihood ratio test $L(x) > K$, and K can be computed for a given x [88, 90, 91].

ROC curves result when disjoint intervals of the random variable x are successively added to the acceptance interval, which is initially empty and at the end contains every value of x . For a given rate of false alarm, the probability of detection is maximized. The expected value of Equation 5 follows easily if the probability of false alarm $P_f = P(Y|n)$ and the probability of detection $P_d = P(Y|s)$ is known. Similarly for the Neyman-Pearson test, the P_d and P_f values completely specify the test performance. Let the cutoff occur at $L(c)$. The hit rate and false-alarm rates are then given by:

$$P(Y|s)_c = P(X \geq c|s) = \sum_{x=c}^{\infty} P(x|s) \quad (8)$$

$$P(Y|n)_c = P(X \geq c|n) = \sum_{x=c}^{\infty} P(x|n) \quad (9)$$

where c is the smallest integer for which $L(x)$ equals or exceeds $L(c)$. A *proper* ROC curve is always concave downward, monotonic and increasing. An example of likelihood ratio ordering is given in Table 1 and Figure 2. To connect points of a discrete distribution on the ROC curve, a mixed decision rule can be used. For example (from Table 1), $P(Y|s) = \gamma_2 P(x = 2|s) + \gamma_1 P(x = 1|s) + \gamma_0 P(x = 0|s) = d$, and by choosing suitable values for γ_i , a value for d can be found. A similar equation can be found for $P(Y|n)$, and the combination of values can be chosen to fall anywhere on the connecting lines.

x	$P(x n)$	$\sum P(x n)$	$P(x s)$	$\sum P(x s)$
2	.3	.3	.6	.6
1	.1	.4	.3	.9
0	.6	1.0	.1	1.0

x	$L(x)$	$P(L(x) n)$	$\sum P(L(x) n)$	$P(L(x) s)$	$\sum P(L(x) s)$
1	$\frac{.3}{.1} = 3$.1	.1	.3	.3
2	$\frac{.6}{.3} = 2$.3	.4	.6	.9
0	$\frac{.1}{.6} = \frac{1}{6}$.6	1.0	.1	1.0

Table 1: Example of the effect of ordering based on likelihood ratio. See also Figure 2

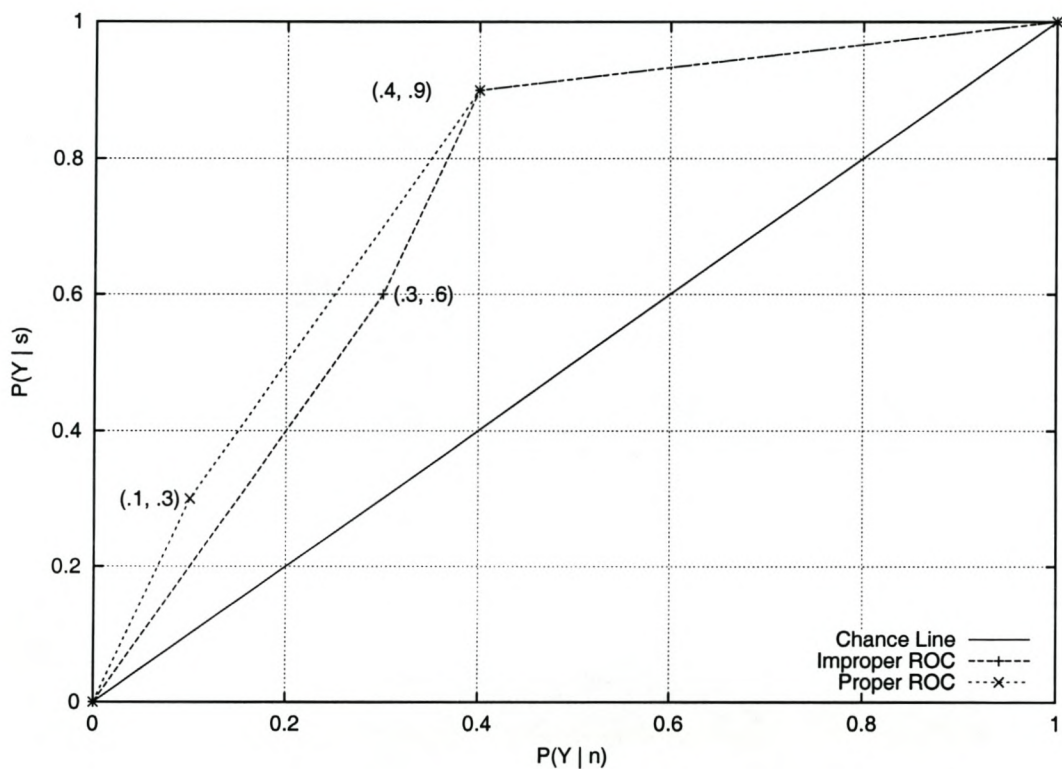


Figure 2: Choosing to order by the likelihood ratio yields optimal performance. The ROC curve is based on the discrete distribution of Table 1, with the proper curve giving the optimal probability of detection for the given false alarm rate. Note that the points on the straight line connecting points on the ROC can be reached with a mixed decision rule.

5 DEADLINER: Building a New Niche Search Engine

5.1 Introduction

Specialised search engines are one of the most powerful and exciting developments in search engine technology today, availing users to specialised information unobtainable with traditional general purpose search engines. DEADLINER is a niche search engine that catalogs conference and workshop announcements, and currently indexes on speakers, locations, dates, paper submission (and other) deadlines, topics, program committees, abstracts, and affiliations. Complex searches can then be performed on these fields, providing information instead of the usual topical relevance. DEADLINER was prototyped using a rapid implementation methodology for specialised search engines. This methodology uses Bayesian integration to avoid the complex and time-consuming manual integration of simple text extraction solutions (or, alternatively, natural language processing), yielding a search engine with an intuitive and rigorous performance control for each field [12].

DEADLINER monitors the World Wide Web, newsgroups and broadcast e-mail for conference announcements. The extracted metadata is stored in a structured database and presented to users via an interface. The interface allows the user to construct complicated queries using extracted fields. Figure 3 displays some of the capabilities of DEADLINER.

A niche search engine is aimed at a specialised web community, with DEADLINER a primary example. Search engines such as Google (Section 2.5) attempt to index all of the Web. Niche engines, on the other hand, scan the Web with the goal of indexing only a small subset of documents relevant to the specialised community. Customised search engines are built on a constrained domain, from which follows that the documents in the community have common elements. These common elements provide the primary motivation for building special-purpose search engines, being the structural concept that makes construction possible. Modelling and extraction of these elements, or exploiting *domain knowledge*, allows complex queries, and these queries can be implemented by taking advantage of the refined web community. Specialised search engines can also be used as information oracles by user agents. For example, DEADLINER can serve as a basis for an agent

that notifies researchers of events relevant to a certain topic at a venue close by. Other examples of non-trivial and highly successful niche search engines aimed at the academic community are ResearchIndex and CORA. Both search engines extract and analyse citation fields from online publications, and perform full page content analysis [67, 58].

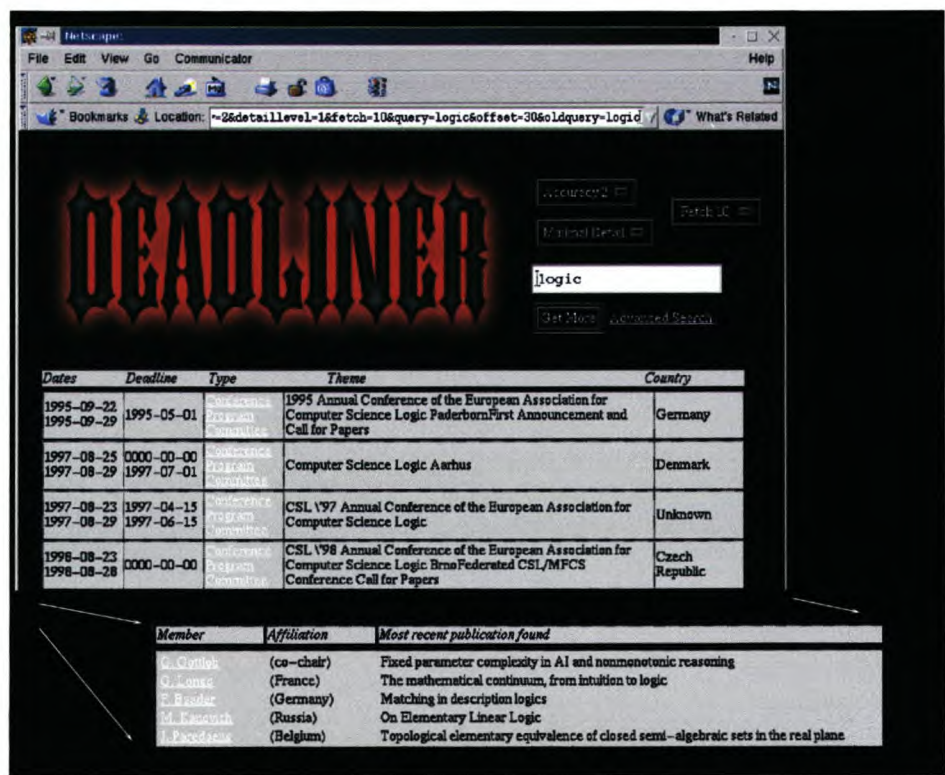


Figure 3: A composite of two DEADLINER interfaces displaying fields automatically extracted from conference announcements. The first (and main) window shows deadlines and themes. The second window shows program committee members and their affiliations.

There are three major obstacles to creating niche search engines. Creating reliable mechanisms capable of detecting relevant documents, extracting target elements from a wide range of sources, and locating relevant documents over the distributed, heterogeneous Web information environment. For example, in DEADLINER, conference related fields must be detected in documents of widely different formats, while ResearchIndex requires extensive knowledge of citation formats for citation extraction. The difficulty of this task, and the resources required, may make it impossible for small communities to construct systems such as DEADLINER or ResearchIndex. Similarly, locat-

ing relevant documents may be beyond the ability of the small community, if done with traditional methods. Section 2.4 discusses methods for alleviating the massive resource investment necessary with traditional information location. The main objectives behind constructing DEADLINER are to provide functionality that could simplify some of the tasks researchers face, and, more importantly, to investigate suitable methodologies for rapidly building specialised search engines. Ultimately, our goal is to build a simple toolkit that can serve as a framework for rapid implementation by a small community. This section therefore introduces DEADLINER and documents a novel methodology for building niche search engines.

We believe that documents on the Web often contain enough structure (e.g. formatting information, link structure and keyword fields) to allow target field extraction without using natural language understanding. This belief is reflected in the methodology inherent to the architecture of DEADLINER. We favour machine learning, rather than hand-tuning, of the system, since machine learning is usually more flexible than the manual approach. Instead of encouraging a large, monolithic solution, the architecture emphasises the integration of multiple simpler partial solutions for extracting text fields. Solutions with various degrees of sophistication can readily be produced by different individuals, which would spread the workload. Comparisons with other methodologies are given in Section 3.6.

The outline of this section is as follows. In Section 5.2, we provide an overview of the search engine, and we describe the architectural challenges. Each of the major components of our general methodology is described as follows: Section 5.2.1 describes document retrieval, Section 5.2.2 describes the use of Support Vector Machines for pre-screening (filtering) of documents for relevance. Section 5.2.3 details detection and Bayesian integration of multiple binary detectors for the detection of target fields, such as deadlines or titles. Section 5.2.4 sets operating points and presents results to the users. The filter extractors used in DEADLINER are described in Section 5.3, followed with a performance evaluation in Section 5.4.

5.2 General Architecture Overview

We produce the architecture of the specialised search engine from the methodology (Figure 4) as follows. First, different retrieval mechanisms locate documents on the Web. A second stage pre-screens the documents for relevancy, discarding and preserving as necessary. Relevant documents are forwarded to a third stage, where multiple extraction filters detect and subsequently extract target fields from web documents. Target fields are stored in a database. We considered different types of databases (e.g. a hash database, and also a simple file structure), but settled on SQL. SQL provides the most flexibility and scalability, especially if future restructuring becomes necessary. We combined the SQL database with a simple hashing scheme on terms (words, keywords) to increase the efficiency of the database lookup, and thereby decrease the latency of SQL lookups.

The huge number of varying formats of web documents leads naturally to the application of appropriate machine learning techniques that can integrate modular extraction filters. The major distinguishing property of the system is, therefore, the focus on automatically integrating many extraction filters for each target field. The cost in terms of complexity and reliability in constructing monolithic solutions (e.g. submission deadlines) proved to be high. Postulating multiple simple solutions, is, however, much easier. These solutions are of different specificity, built from regular expressions, and are well suited to extracting target fields from disparate formats. The Bayesian approach we provide automatically integrates the simple regular expression filters in an optimal fashion using relatively modest amounts of labelled data. Improved performance over a single extractor/filter can be achieved, since the integration phase exploits the joint statistics across the different filters.

A second major problem addressed by the Bayesian approach is that of choosing appropriate probabilities of detection and false alarm, i.e. selecting an appropriate operating point. With a visual display of results it is easy to display the results in decreasing order of likelihood; but for an e-mail based application activated by an extracted date, users would want to indicate a different acceptable false alarm probability. A too high probability of false alarm would cause unnecessary generated email. A too low probability of false alarm, and the system would be not be useful. Fortunately,

the Bayesian integration approach discussed in Section 5.2.3 leads to a mechanism for varying the operating point for each user, even though the operating point of each simple extraction filter is fixed. Performance is controlled by a single parameter, and is monotone. The system can also evaluate the effectiveness of new filters as they are added, and integrate them as appropriate.

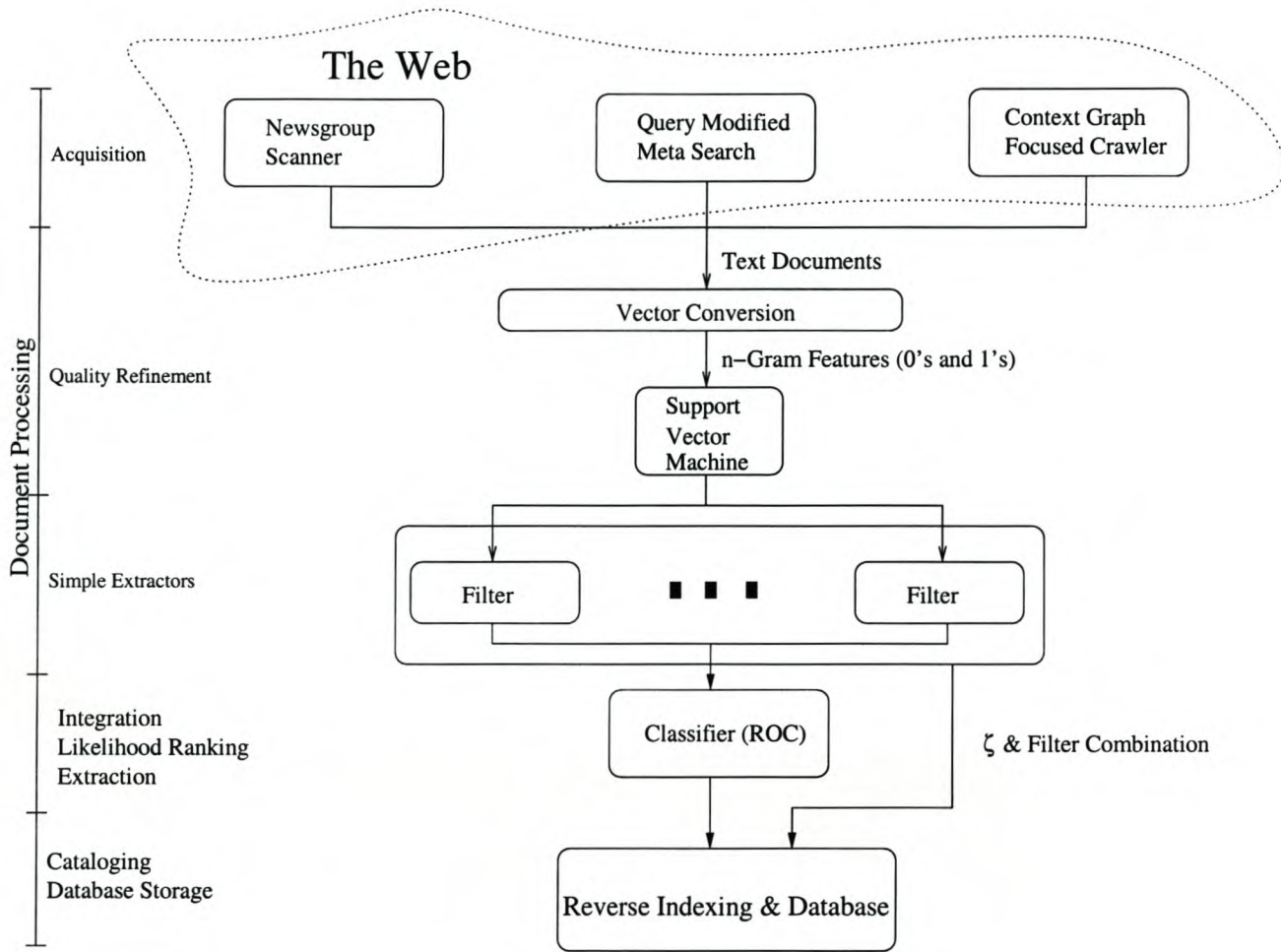


Figure 4: The basic framework of our specialised search engine implementation. A primary (SVM) classifier selects individual documents for further processing. A secondary classifier assigns posterior threshold ratios $L(y)$ to every document and extracts target fields. The user queries with the database by selecting a specific operating point for accesses, and providing standard queries, such as a date or a keyword.

5.2.1 Stage I: Document Retrieval

Crawling the Web exhaustively is not feasible. Since we are interested in only a small percentage of the Web pages, the cost of crawling the Web this way would be prohibitive. Also, a significant portion of the information in DEADLINER needs to be current. We use multiple lines of attack on this problem.

A simple, yet effective strategy uses simple polling scripts that download artifacts from well-known sources where seminar and conference related materials commonly appear, including newsgroups, universities and professional organisations.

Traditional crawlers need a significant amount of resources, especially since the breadth-first search strategy is followed. We decided to use the Context-Graph Focused Crawler developed by Diligenti *et al.* to extend our grasp. This crawler uses the link structure and contents of documents to improve the retrieval rate of documents related to the training set. See Section 2.4 for more detail on this and other crawlers [14].

Commercial search engines form the base for the last input module towards finding relevant documents. We extracted query modifiers (a list of keywords and short phrases) from the training data. The query modifiers are periodically submitted to the metasearch engine Inquirus 2. Inquirus 2 supports query modifiers, and in turn queries approximately ten commercial search engines, allowing leveraging of commercial search engines. We refer the reader to Section 2.6, and to detailed descriptions of this work available in [4, 57].

5.2.2 Stage II: Pre-Screening Using SVMs

The Web documents obtained in the previous section are pre-screened with support vector machines (Section 3.5). The methods used above provide some degree of relevancy, but the documents are still not of a sufficiently high quality. To improve the *precision* of the documents we employ SVMs. SVMs has been shown to work excellently with vectors of high dimensionality and are resistant to over-fitting. The downloaded documents are therefore pre-screened with an SVM

built for text classification (Section 3.5). We used the standard sequential minimal optimisation SVM algorithm described by Platt. We refer the reader to Kwok, and Section 3 for discussions comparing SVMs to several different text classifiers, and to Haykin for a technical description. We next detail the construction of the feature vectors used as input to the SVM used for pre-screening [92, 93, 94, 3, 78, 77].

We use short phrases as features (such as a word, a bi-gram, a trigram, where, for example, “international conference” is different from “conference international”) to construct the vectors used by the SVM. A document can then be represented by a feature vector. A feature vector is written as a list of present/not present labels, combined with the class label. The class label is positive or negative, depending on the class of the document. The training data generates thousands of possible features, and to reduce memory usage and increase efficiency, we reduce the SVM feature vector size by selecting the most important features. If a feature does not occur in more than 7.5% of either the true or false classes, we prune that feature from the set of features considered. The candidate features are then ranked in order of the ratio of their frequency on the true class, to their frequency on the false class documents. The top N (typically 100 to 300) candidates are then used as features for the input vectors for the SVM classifier, and we finish by training the SVM on these feature vectors.

Unseen documents are converted into the reduced size feature vector, and the SVM indicates relevancy after evaluating the vector.

5.2.3 Stage III: Bayesian Detector Fusion

A major problem in extracting target fields is to locate suitable document cuts for the application of extraction rules. Extracting information blindly from large portions of documents results in irrelevant or simply incorrect extraction of information. To address this problem the document is first cut into blocks or sentences, and then the document cuts are submitted to a reliable relevancy detection mechanism. We follow detection with extraction performed on the cuts that show a high likelihood of relevancy.

To detect a specific target, we construct a detector by optimally integrating a number of simpler detectors for every target field. Suppose we wish to extract the conference start date, and that we are using a set of regular expressions and formatting rules. We call these elements *filters*, and associate a binary class variable with each detector that indicates whether the filter matched or not. The combination of filter and match variable we denote with *detector*, written as f_i . How do we integrate these partial detectors? We need a detector whose performance exceeds that of any of the constituent detectors. Furthermore, it must be possible to easily change the operating point (precision/recall setting), even though the constituent filters are all fixed. To achieve our objective, we combine the simpler detectors using the Bayesian approach and the Neyman-Pearson procedure to yield an optimal classifier.

Applying the simple detectors to the text input space X yields either “yes” or “no” responses. For this detector/classifier, we wish to find the combination of detector/classifiers with the highest probability of detection (or “hit rate”) for a given rate of false alarm. Formally, let $B = \{0, 1\}$, represent the possible target output from a detector/classifier with 1 indicating a detection and 0 otherwise. The detector/classifier can then be written as $f_i : X \mapsto B$. We would like to have one classifier $C : X \mapsto B$. To utilise the simpler classifiers f_i , we construct C as $\Gamma \circ f_v$ with $f_v = (f_1, f_2, \dots, f_N)$, the n -tuple of binary outputs of the f_i ’s, and $\Gamma : B^N \mapsto B$. That is, $C : \lambda \circ f_v \mapsto B$. Since the range of C is of cardinality 2, and the domain is of cardinality 2^N , the Cartesian product yields 2^{2^N} different possible combinations. The domain of C can be seen as a bit string, with every combination of elements in B^N mapped with the customary mapping to the natural numbers $\mathcal{N} = 0, 1, \dots, N - 1$. Rewriting C , we have $C : [0..N - 1] \mapsto B$. Section 4 explains the application of Bayes’ criterion and the Neyman-Pearson procedure to discrete distributions, and provides a small example using natural numbers ($\mathcal{N} = 0, 1, \dots$) as parameter.

We define, similar to Section 4, the likelihood ratio $L : X \mapsto \mathbb{R}^+$

$$L(x) = P(x|H_1)/P(x|H_0), x \in [0..N - 1] \quad (10)$$

We define the probability of detection and the probability of false alarm, with H_0 indicating an irrelevant signal, and H_1 indicating a relevant signal or match. The bit strings that fall in the relevant

class we denote with $A_\Gamma = \{y | \Gamma(y) = 1\}$.

$$\begin{array}{l|l}
 P_d(\Gamma) = \sum_{y \in A_\Gamma} P(y|H_1) & P_f(\Gamma) = \sum_{y \in A_\Gamma} P(y|H_0) \\
 = \sum_{y: \Gamma(y)=1} P(y|H_1) & = \sum_{y: \Gamma(y)=1} P(y|H_0) \\
 = \sum_y \Gamma(y) P(y|H_1) & = \sum_y \Gamma(y) P(y|H_0)
 \end{array} \quad (11)$$

An ROC curve is the set of operating points where, for a given P_f , we find the maximal P_d . We note that there are 2^{2^N} possible combinations of $R_\Gamma = \{P_f(\Gamma), P_d(\Gamma)\}$ pairs. The ROC curve provides a measure for distinguishing between the two possible hypotheses H_0 and H_1 . Comparing every possible combination of operating point R_Γ is not practically possible except for very small values of N . Since we are using the Neyman-Pearson design procedure, we have a ranking of the possible 2^N strings according to the likelihood function L , which yields an exponential decrease in complexity. The classifiers that lie on the ROC are sufficient statistics for building the ROC. By assigning the 2^N strings in decreasing order of likelihood ratio to the true class decision region, the ROC support classifiers are found in order of increasing false alarm performance. Thus, we have 2^N ROC support classifiers [95].

For example, assume that two binary features $y = (y_1, y_2)$ correspond to matching two regular expressions. The true class distributions of H_0 and H_1 yield real values $L(y)$ for each combination of these features. By ranking $L(y)$ over all y in decreasing order, we obtain the y , and therefore the exact values for (y_1, y_2) for which we compute the corresponding P_d and P_f values. Different operating points would require using different regular expressions, or, a combination of regular expressions.

The operating points produce results of different accuracy. Since users' needs may differ, we note that a classifier Γ^j is created by labelling all feature combinations or strings y (i.e. histogram bins) whose likelihood ratio $L(y) \geq L_j$, where L_j is the j th ranked likelihood value, as relevant. When processing the outputs from a classifier applied to a text input, we do not pick a specific classifier and operating point up front. We map every feature combination y to its likelihood $L(y)$. Since we store $L(y)$ in a database, we can then later easily choose any detection operating point by choosing an appropriate threshold for the likelihood values. In this way, every user or agent can maintain a

different operating point by varying a single personal threshold, and lookup of the data is quick.

Estimating *a priori* probabilities on the Web is difficult. An advantageous property of the Neyman-Pearson design is that it is robust to changes in *a priori* probabilities, in the sense that the operating point may move along the ROC, but the set of classifiers do not change. One could simply change the user threshold to account for changes in these probabilities, and re-classification (of stored data) would not be necessary.

Another noteworthy point is that the Neyman-Pearson design approach is a search procedure that results in a big reduction in complexity. Finding the classifier function Γ that maximises the P_d at a given value of P_f is reduced from searching a space of dimension 2^{2^N} to one of 2^N .

Finite Data and Feature Subset Grouping The true class conditional distributions are unknown in practice, since we do not have infinite amounts of data, and the training data might be skewed. We estimate statistics from a finite labelled data set. Ensuring accurate histogram estimates is non-trivial, and a major subject of investigation. For detailed analyses on estimating the performance curve and an investigation into the sensitivity of the likelihood ranking procedure, data set size and number of filters, we refer the reader to Coetzee *et al* [96].

5.2.4 Stage IV: Presentation and Cataloguing

We described a method for combining the outputs of simple detectors that allows us to detect a region of text that could possibly match a target element. Extracting relevant fields from these regions is the next problem we should address to create a functional niche search engine. Every extractor generates a binary feature on a match (0 or 1), and an actual estimate of the value of the target element (the text block that triggered the match). An overall detection operating point can be set for every target element. A setting can overrule some of the filters, or require a certain combination of filters. The text fields extracted by the filters that both triggered a match and have a positive weight in the integration are merged. To extract we use heuristics, e.g. by using the smallest common text segment, or by identifying tokens, such as dates, and then using the smallest

common text segment.

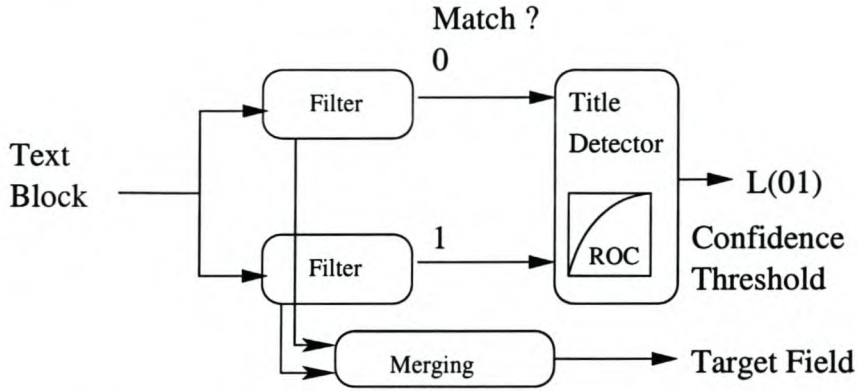


Figure 5: A number of different filters attempt to recognise the desired target in the cataloguing section shown here. Every match of a filter results in a binary feature. Every match is integrated with responses from other detectors. Every combination of filter matches y can be mapped to an associated threshold $L(y)$. The value of $L(y)$ reflects a detection confidence, which is stored with every field value obtained by merging the fields of the active filters.

We process single paragraphs or sentences at a time. Multiple matches refer to the same text segment. Target elements can be detected at different locations in the document, when considering long paragraphs or the document as a whole. Even though there are restrictions on the fields, we cannot always force these restrictions without losing information. For example, even though titles almost always occur only once, there are documents on the Web that contain multiple documents pasted together to form one big document, yielding more than one relevant title in the big document. To compound this problem, the data extracted from the matches may conflict. Every text segment has a confidence estimate $L(y)$, provided by the integrator. We use the match with the highest confidence, since multiple matches are possible. The confidence values may be very similar, thus, we index on all the matches that have equal or almost equal values. Incorrect results may be returned from queries, but in principle we do not make irreversible decisions.

5.3 Filters used in DEADLINER

We developed an architecture in the previous sections that can easily be ported to different domains by developing domain specific extractors (regular expressions), and by labelling a set of training data. This section describes these domain specific components of DEADLINER.

5.3.1 Features and Filters

Inspection of a number of document sources visually gave a very good indication that most conference materials follow a block layout. We found that the following blocks occur frequently: the title, a listing of affiliations, abstracts, organisers such as a program committee, discussion topics, venue, scope and objective statements, and a miscellaneous information section (describing topics ranging from the weather to local social events). Filters produce excellent results without extensive natural-language processing, since the block structure reduces the complexity of the extraction problem.

The simplest filters in the system perform keyword and short phrase matching, based on a vocabulary generated using word frequencies. We also make extensive use of databases of lists of authors (extracted from ResearchIndex), lists of research areas and keywords (extracted from ResearchIndex) and venue and geographical names.

A number of simple primitives are constructed with these databases. The primitives are then used to construct filters that utilise constraints based on the block structure. Table 2 shows some of the filters that are currently implemented. The filters are aimed at extracting common elements of conference announcements, namely the title, deadlines, lists of topics and program committee members. The filters are described using an extension of standard regular expression notation that allow for formatting and database lookup: $*$ means zero or more (Kleene closure), $+$ means one or more, $\langle \textit{description} \rangle$ indicates information obtained outside the regular expression, or is used for clarity. We use ω for the set of alphanumeric characters, and ς for whitespace (newlines, tab-stops, spaces). A range of characters is denoted by $[]$ (e.g. $[A - Z]$ denotes capitals). The

function *indentation(text)* returns a list of numbers indicating the level of indentation for each line in the text, while *max(list)* and *min(list)* returns the maximum and minimum of the numbers in the list respectively. The operators *country_names* reflects presence in a list of country names, and *known_names* a match in a list of about 90,000 proper names. We use the notation *<regularly_occurring_separators>* for separators that occur most frequently in a list. The separator may contain whitespace, but does not consist only of whitespace, and is always preceded by a newline. A **B** indicates a word/non-word boundary, and *..* is used to indicate a range. A *|* means *or* (unification), while *()* indicates grouping. The operator *!* negates a character set, while a full stop *(.)* matches any character except a newline. Occurrences of a symbol of least *n* times but not more than *m* times is denoted by *{n, m}*, while *#(<expression>, <expression2>)* counts the number of times *<expression2>* occurs in *<expression>*. In *(<expression>, <expression2>)* the part *<expression>* is the index of the text block, while *^* is used to anchor the beginning of text.

Interestingly, the title is much easier to detect if the word *on* is present. Though the word is a common word in normal vocabulary, the combination with the other features increases the probability of detection significantly. The deadlines are mostly identified with keywords occurring in and around the important dates³⁷. The indentation is a major component of detecting a list of topics, since topics are very different between announcements. Lastly, the program committees exhibit regularity in their formats, allowing the simple filters.

Constructing fixed lower level filter parameters using such simple heuristics is at the center of our motivation. Diversifying over the filters allows different operating points, preferable to the difficult problem of varying internal filter parameters.

5.3.2 Heuristics

Following detection of a target element, we can use heuristics to combine the text fields extracted by the active filters. For example, a detector might return a title with the sponsors' information

³⁷One of the well-known problems with extraction of dates is the inherent ambiguity of many of the date formats (e.g. 92111).

attached, and this information should be removed. We briefly describe three elements to provide an indication of the complexity of our approach:

Program Committees: The text obtained from the detectors is split into tokens and matched against a dictionary of known author names and possible affiliations assembled from ResearchIndex papers (there are approximately 90,000 distinct dictionary elements), then against a list of common dictionary words, and ultimately a dictionary of country and place names. All matching words are replaced by symbols denoting the dictionary in which they were found. We then use regular expression templates to match the resulting symbolic strings. A regular expression is then constructed for all the positive matches, and re-applied to the text to find persons not in the dictionary.

Deadlines: We match standard date formats in sentences and tables. Extraction of the surrounding or immediately preceding text is used to determine the type of deadline (e.g. abstract submission date).

Titles: A title usually contains at least two of: (i) country name, (ii) city/state name, (iii) date of meeting, (iv) deadline, (v) list of sponsors, (vi) name, (vii) acronym for the conference, and (viii) theme/summary of the conference. We enforce the presence of at least two elements. Most elements are recognized by matching entries from a database. The summary is generated by removing elements (i) through (vii) from the match.

Name	Primitive	Target	Meaning
0	(<i>cur.par.</i> , /\$kwd/)	title	match keywords in current paragraph
1	(<i>cur.par.</i> , /BonB/)	title	match "on" in current paragraph
2	#(<i>cur.par.</i> , / $\omega^+\varsigma^+$ /) <= 30	title	count the number of words
3	#(<i>cur.par.</i> , /[A - Z] $\omega^+\varsigma^+$ /) / #(<i>cur.par.</i> , / $\omega^+\varsigma^+$ /) >= 0.75	title	ratio of capitalized words:words
4	min(indentation(<i>cur.par.</i>)) \neq min(indentation(<i>prev.par.</i>))	title	difference in indentation
5	#min(indentation(<i>cur.par.</i>)) \neq min(indentation(<i>next.par.</i>))	title	difference in indentation
6	(<i>cur.par.</i> , /<date> /)	title	match a date
7	(<i>cur.par..cur.par</i> + 5, /<date> /)	title	match a date in the next 5 paragraphs
8	(<i>cur.par.</i> , <= 7)	title	this is one of the first paragraphs
9	(<i>cur.par.</i> , /<countryname> /)	title	match a country name
10	#(<i>cur.par.</i> , /[$\omega\varsigma$] + /) > 20	title	count the non-whitespace, non-alphanumeric characters
0	(<i>cur.sentence.</i> , /deadline/i)	deadline	match the word "deadline"
1	(<i>cur.sentence.</i> , /by before due closing/i)	deadline	match "by" or "before" case insensitive
2	(<i>cur.sentence.</i> , /later ς^+ than/i)	deadline	match "later than" case insensitive
3	(<i>cur.sentence.</i> , /on/i)	deadline	match "on" case insensitively
4	(<i>cur.sentence.</i> , /<date> /i)	deadline	match a date
5	#(<i>cur.sentence.</i> , /<date> /i) \geq 3	deadline	there are three or more dates
6	#(<i>cur.sentence.</i> , /<date> /i) > 0	deadline	match a date
7	(<i>cur.sentence.</i> , /submi/i)	deadline	match "submi"
8	(<i>cur.sentence.</i> , /paper/i)	deadline	match "paper"
9	(<i>cur.sentence.</i> , /:/i)	deadline	match ":"
10	(<i>cur.sentence.</i> , /notify notification accept camera/i)	deadline	match "deadline qualifiers"
11	(<i>cur.sentence.</i> , /important ς^+ date/i)	deadline	match "important date"
0	(<i>prev.par.</i> , /\$kwd/)	topic	match a keyword
1	(<i>prev.par.</i> , /:/)	topic	match a colon
2	(<i>prev.par.</i> or <i>cur.par.</i> , /<regularly.occuring.separator> /)	topic	is there a regularly occurring separator?
3	(<i>cur.par.</i> , /<regularly.occuring.separator> /)	topic	is there a regularly occurring separator?
4	#(<i>cur.par.</i> , /. /) / #(<i>cur.par.</i> , /<line> /) < 0.1	topic	ratio of full stops:number of lines
5	(<i>cur.par.</i> , /\$kwd/)	topic	match a keyword
6	(<i>cur.par.</i> , /limited) or (<i>cur.par.</i> , /limited ς^+ to/i)	topic	match "limited to"
7	(<i>cur.par.</i> , /includ/i)	topic	match "includ"
8	min(indentation(<i>cur.par.</i>)) = min(indentation(<i>next.par.</i>))	topic	minimum indentation differs
9	(<i>cur.par.</i> , /interest/i)	topic	match "interest"
10	max(indentation(<i>cur.par.</i>)) = max(indentation(<i>next.par.</i>))	topic	maximum indentation differs
0	(<i>cur.par.</i> , /\$kwd/)	program committee	match the keywords
1	(<i>cur.par.</i> , /univ/)	program committee	match "univ"
2	(<i>next.par..next.par</i> + 1, /univ/)	program committee	match "univ"
3	(<i>cur.par.</i> , /<countryname> /)	program committee	match country names
4	(<i>next.par..next.par</i> + 1, /<country.names> /)	program committee	match country names
5	(<i>cur.par.</i> , /<known.names> /)	program committee	match known names
6	(<i>next.par..next.par</i> + 1, /<known.names> /)	program committee	match known names
7	(<i>cur.par.</i> , /[$\omega\varsigma$] + /)	program committee	match non-whitespace, non-alphanumeric chars
8	(<i>next.par..next.par</i> + 1, /[$\omega\varsigma$] + /)	program committee	match non-whitespace, non-alphanumeric chars
9	#(<i>cur.par.</i> , /B[A - Z]B/)	program committee	number of single letter words
10	(<i>next.par..next.par</i> + 1, /B[A - Z]B/)	program committee	number of single letter words

Table 2: Primitive operators used for constructing filters. Each primitive is aimed at extraction of a particular target concept. We also associate a specific area of text with each filter.

5.4 Performance

SVM Performance: The data used to train the SVM in Section 5.2.2 was a set of 592 manually classified Calls for Papers (CFPs), and 2269 negative examples, consisting of several “random” URLs from the Inquirus 2 logs, and approximately 850 conference related pages. A good CFP contains a title describing the event, a list of topics, a program committee, deadlines and submission information. CFPs were collected by combining URLs from documents containing lists of CFPs, and by combining searches from various search engines (For example, search for “Calls for Papers” in a normal search engine).

The training set consisted of 249 positive and 1250 negative sample documents, which were randomly selected. We limited pages from any one domain to 20, to prevent a bias in unseen documents. The remaining 343 positive and 1019 negative exemplars formed the test set.

Table 3 summarizes the results on the test set. We obtain excellent results using extremely limited structural processing. From our evaluation of the test set, we noted that non-English or multilingual sites represent a major problem for our system, expected from the biases of our dictionaries. A Gaussian kernel in the SVM produces noticeable improvements over a linear SVM, but the significant extra overhead does not warrant use with the hardware we have at present. Consequently, we decided to use the linear classifier for evaluating web documents.

Collection	Type	No. Pos	No. Neg	Pos. Accuracy	Neg. Accuracy
CFP Test	Gauss	343	1019	95.9%	98.6%
CFP Test	Linear	343	1019	88.1%	98.7 %

Table 3: Summarized results for the SVM call for paper classifier. There was no overlap with the training set.

Extractor Performance: We used 500 documents from DBWorld³⁸ for training, and 100 documents from DIKU³⁹ for testing of the feature extractors. The Web is extremely diverse, and ob-

³⁸<http://www.cs.wisc.edu/dbworld/>

³⁹<http://www.diku.dk/research-groups/topps/Conferences.html>

taining representative data on the Web, is difficult. We therefore made the deliberate decision to test the fragility of our detectors/extractors by using a different source for our test data, so that the class distributions of the training and test data would differ significantly.

These data sets were labeled to indicate desired target fields. The DBWorld documents contain 208 lists of interesting topics, 338 conference titles, 906 deadlines and 197 program committees. We chose not to exclude announcements that are not strictly calls for papers, since our SVM classifier misclassifies a small percentage of web documents, and we therefore need fairly robust detectors.

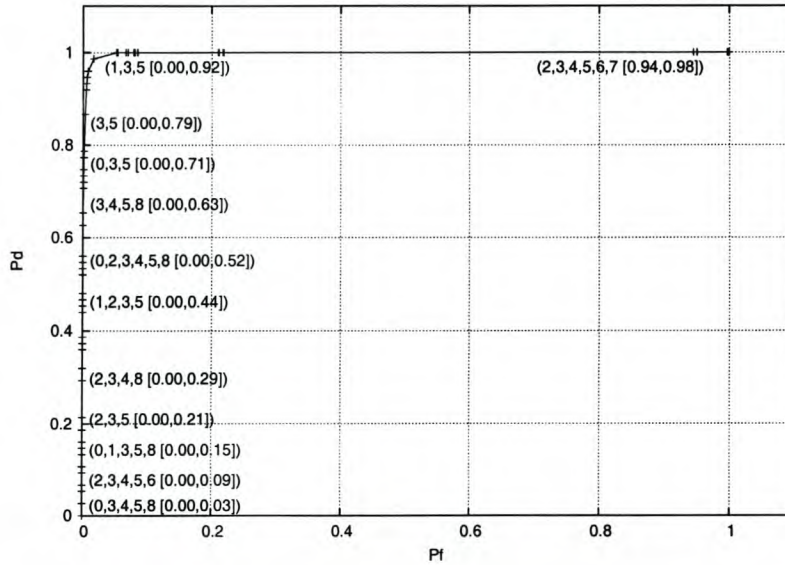


Figure 6: Detection performance of the detector for the target field “deadlines”. Some of the optimal detectors and associated operating points (P_f , P_d) that are used to construct the performance curve is shown. The integers indicate the combination of detectors from Table 2 used to obtain an operating point. The curve is statistically accurate on the training data; only a few of the detector combinations is shown, to prevent clutter.

Figures 6–9 contain the ROC curves for different integrators. The integrators detect the deadlines, conference topics and program committee members, and were obtained from the DBWorld dataset. These curves are statistically accurate, on the condition that the class distributions of new data matches that of the DBWorld data. We show the performance obtained by combining multiple detectors. In each case a performance improvement results from integrating multiple detectors.

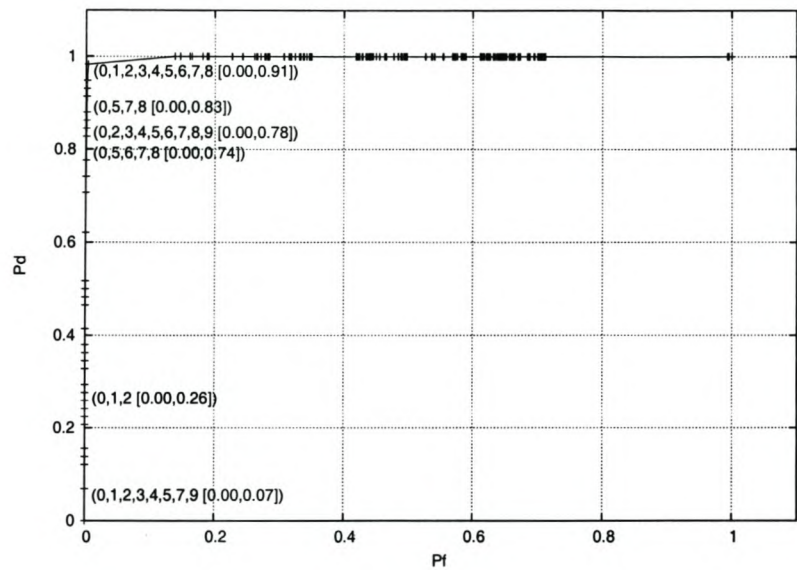


Figure 7: Detection performance of the detector for the target field “program committee”.

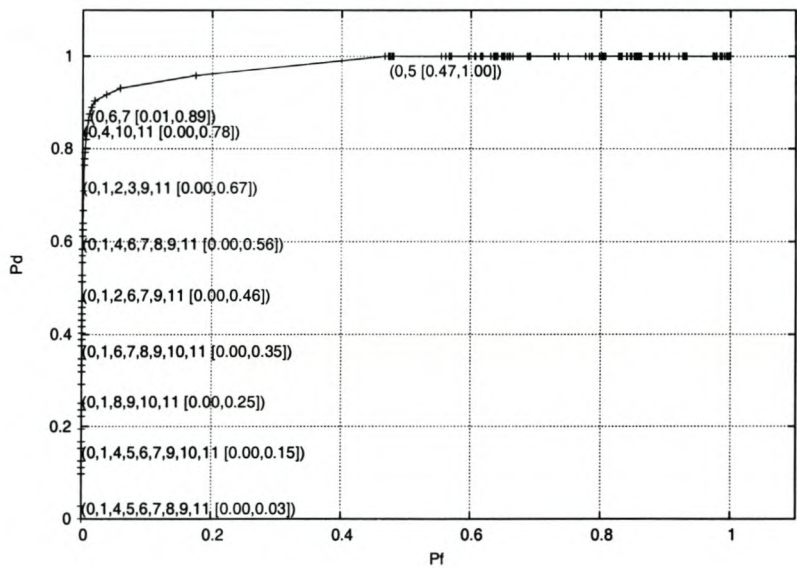


Figure 8: Detection performance of the detector for the target field “topics”.

The values show that different combinations of features are preferred in different ranges, although some filters are always desirable.

After detection of the target field, the actual target field value was extracted using heuristics, or with the actual detector, since many of these detectors were chosen such that extraction would

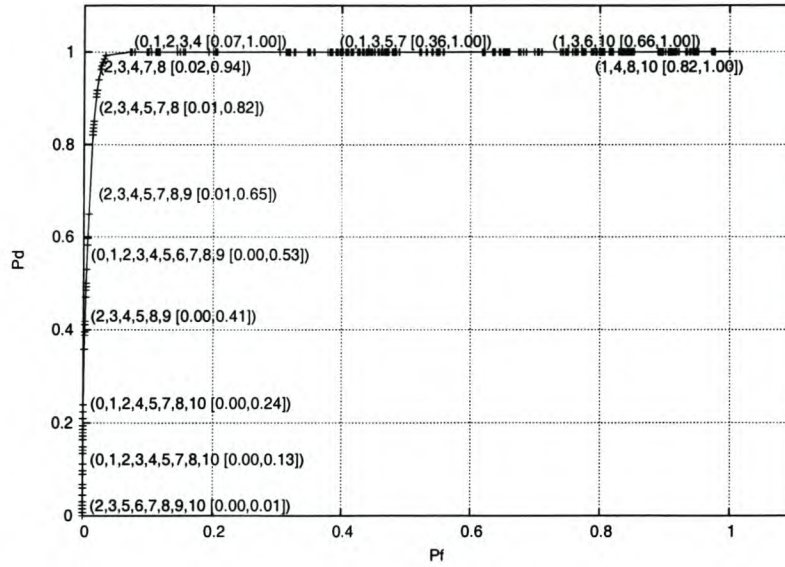


Figure 9: Detection performance of the detector for the target field “titles”.

follow automatically after detection. These results are contained in the DIKU data, and are shown in Tables 4–5.

The results on deadline extraction is shown in the first row of Table 4. The target rate of P_f was chosen around 5% for the operating point. The document set contained a total of 300 deadline date fields. Of these, 214 deadlines were correctly detected and extracted, and 2 dates were detected, but incorrectly extracted. A total of 31 non-deadline dates were detected and extracted (usually dates from a program announcement), and is listed as extraneous. These dates will be indexed, but the user can resolve the ambiguity without much trouble. We required that the text that describes the deadlines had to be perfectly extracted: 86% of the time the text was considered to be correct. Our overall deadline extraction accuracy is therefore approximately 70%. By evaluating the errors, we found that errors usually result when dates are given in tables and also, many highly irregular date formats were used (for example: 1 & 2 & 3 January 1989); we aim to develop new parsers to improve processing in this domain.

The second row of Table 4 summarizes the performance on the program committee extraction task. Of the 1455 program committee members, we found 1252 with our system, performing at

Target	Total	Detected/Extracted	Detected/Not Extracted	Extraneous
Deadline	300	214	2	31
Committee & Affiliation	1455	1252	72	136

Table 4: Extraction results for target concepts “Deadline” and “Program Committee”.

87% accuracy.

Various title components’ extraction performance is shown in Table 5. Since title composition varies widely, we provide these results as percentages. While we expected difficulty with the date fields (the start and end dates were confused many times, and the notation used in announcements is non-standard), we had relatively disappointing performance on country and city names. An expansion of our dictionaries should improve performance. An increase in performance was achieved in extraction of the theme and name of the conference, and in identifying the type of conference. We achieved almost 90% in accuracy with the conference names and the type of meeting.

Start Dates	End Date	Theme/Name	Country	Type of Meeting
73%	71%	81%	77.5%	85%

Table 5: Extraction results for target concept “Title”, with associated subfields.

There is a prominent increase in false detection when using the DBWorld filters and extractors on the DIKU data collection, with the error usually doubling from the expected value. The style differences between DBWorld and DIKU are the main cause for the dissimilarity in the underlying distributions, and increase the error values. These error values show the challenges inherent in developing web information systems. The challenges include data that accurately reflect the underlying distributions, and the widely varying formats on the Web. Visual inspection of the data in DIKU shows a much different style in presentation of announcement information. The results are, however, adequate for DEADLINER to be a useful tool. Additional labeled data should be available from user feedback, and the labeled data from DIKU will also be used to improved the performance of the system.

5.5 Summary

We presented DEADLINER, a research tool that currently catalogs conference and workshop announcements. DEADLINER extracts deadlines, topics, program committees and titles of these announcements. The extracted information can be used to find relevant conferences and seminars, and allows the monitoring of smaller, local events. Our aim is for DEADLINER to be able to collect and extract a wide range of academic assembly and seminar related materials from the Web, reducing the some of the labour related to research.

DEADLINER was modularly constructed with a general method for the construction of special-purpose search engines, and can easily be reconfigured to create niche search engines for other domains. The architecture resulting from the methodology supports the location, extraction, and cataloguing of domain specific metadata. We believe that most domains on the Web contain enough information in the form of formatting, keywords and phrases, and specialized dictionaries to allow field extraction, thus the form of our architecture. The metadata is then disseminated via a web interface. Locating domain relevant artifacts are aided with state-of-the-art focused crawlers and classifiers, reducing the effort required to obtain these artifacts. Following acquisition, information artifacts are pre-screened with a classifier based on support vector machines, yielding high-quality relevant artifacts. These artifacts are then analysed, and relevant portions are detected with filter/detectors, integrated with an application of Bayes Theorem, and the Neyman-Pearson design procedure. The detected metadata is extracted and catalogued, and, lastly, presented and disseminated to users.

6 Conclusions and Directions for Future Research

6.1 Conclusions

Special-purpose search engines are a new development on the World Wide Web, responding to increasing requirements in quality of information, more powerful search techniques, and inadequate coverage by existing search engines. The definite viability for construction of special-purpose search engines, and the many disparate web communities, are pushing the development of the Web in the direction of special-purpose search services. In contrast to general-purpose search engines, special-purpose search engines can catalogue a large subset, if not the whole set, of relevant artifacts. Almost every text artifact on the Web should be relevant to the larger general purpose search engines, creating technical challenges which might not be easy, or even possible, to overcome. Despite the obstacles, major advantages result from general search engines, amongst which: the full source of an artifact is available, a large number of relevant artifacts are indexed, and generality. However, the cost associated with such a search engine is prohibitive, many of the resources are rarely used, and a low precision and recall results in many queries, since relevance can be difficult to determine. Metasearch engines have big coverage, but does not index the complete source of web artifacts. This results in hard relevance decisions, and fusion of results from different search engines is very difficult. Special-purpose search engines are relatively low in resource cost, domain specific, use structured data, are more up to date, can be successfully personalized, and can make very good relevance judgements based on the full information artifact. Structured queries allow the user to obtain very specific information, thereby reducing the users' time and effort spent in finding relevant information.

An interesting area for research is focused crawling. Unfortunately, there is at the moment no comparative study on the performance of the documented techniques. The performance of most of the techniques are compared to breadth-first search, and performs orders of magnitudes better. One of the techniques uses the information theoretic precision measure; it would be informative to see a comparison of all those techniques. Topic drift does not seem to be a big problem, but is not

investigated for most of the techniques. Furthermore, only learning techniques are applied, except for the first focused crawler which was applied to the very small web of 1994. Machine learning seems then, to be very eminently suited to focused crawling.

Classification of text documents is widely studied, with SVMs performing very well. Locating relevant artifacts, pre-screening done with SVMs, gives good performance. The combination of focused crawlers, query modified search engines, and polling scripts, achieves acceptable performance. An open question is whether good coverage results. The indications are there, and query modified search combined with crawling is intended to increase the recall, but some test is needed to verify this hypothesis.

Information extraction is inherently difficult. HTML tags can alleviate the problem, as do formatting information. The filters developed with DEADLINER fulfill multiple functions, that of detecting relevant portions of a document, and then extracting the relevant information. The acceptable levels of performance creates a useable tool; increasing the quality of training data and adding filters should improve performance, since the optimal integration takes advantage of the underlying distribution of detectors. The basic architecture is general, and applicable to almost any web community.

Most of the constituent technologies necessary for creating a set of tools enabling rapid development of special-purpose search engines are reaching maturity. Building niche search engines is not only viable, but necessary to increase the location and dissemination of relevant information on the Web.

6.2 Directions for Future Research

Adopting the DEADLINER architecture to other domains, and studying its applicability, should prove interesting. At the time of writing there is no comparative study of the focused crawlers discussed in literature. Comparing the different focused crawlers, with one another, and with techniques that leverage existing general-purpose search engines, such as query modification, should

increase the efficiency of search engines such as DEADLINER.

DEADLINER performs optimal integration of detectors, but we are still limited to a fairly small number of filters (about thirty). One improvement would be to find a way of increasing this number significantly, and use a massive number of even simpler detectors. This could reduce the work of the designer, since filters developed for other domains might be applicable, and can then simply be included for evaluation. One possible decision-tree based method that utilizes the information theoretic entropy measure such as C4.5 might be useful, since the tree structure could provide an integration procedure.

The individual detectors are built on manually constructed filters. Even though these filters are easy to construct from keywords and regular expression, the filters still need domain knowledge, and knowledge of regular expressions. Using the idea in the above paragraph, and using a discrete Hidden-Markov Model for a (simple) detector, could reduce the construction time further still, and reduce the need for constructing manual detectors.

A completely different approach that allows efficient search and indexing operations on metadata can be done by changing the underlying structure of web pages. Semantic web⁴⁰ under development at the standards society for the World Wide Web is one such example. The big disadvantage of this approach is that, since the current standard is so entrenched in the way information is represented, it would be very difficult to change to a new standard. Many documents would have to be reformatted or changed, as well as the current methods for indexing that search engines use, which would be a significant economic setback. Continuing to explore metadata creation and special-purpose search engines such as DEADLINER, is therefore advisable.

⁴⁰<http://www.w3.org>

References

- [1] “Bow Tie Press Release,” 2000. http://doc.altavista.com/company_info/press/pr051100.html.
- [2] S. Lawrence and C. L. Giles, “Accessibility of information on the web,” *Nature*, vol. 400, no. 8, pp. 107–109, 1999.
- [3] T. Joachims, “Text categorization with support vector machines: Learning with many relevant features,” in *Tenth European Conference on Machine Learning (ECML-98)*, vol. 44, (Dortmund), pp. 137–142, 1999.
- [4] E. J. Glover, S. Lawrence, M. D. Gordon, W. P. Birmingham, and C. L. Giles, “Web search – your way,” *Communications of the ACM*, 1999.
- [5] B. M. Leiner, V. G. Cerf, R. E. K. David D. Clark, L. Kleinrock, D. C. Lynch, J. Postel, L. G. Roberts, and S. Wolff, “A brief history of the internet,” August 2000. A version of this document appears in the 50th Anniversary edition of CACM, February 1997, <http://www.isoc.org/internet/history/brief.html>.
- [6] D. D. Clark, “The design philosophy of the DARPA internet protocols,” in *Proceedings of ACM SIGCOMM '88*, pp. 106–114, August 1988.
- [7] “A little history of the world wide web,” 1995. The World Wide Web Consortium, <http://www.w3.org/History.html>.
- [8] J. G. Brookshear, *THEORY OF COMPUTATION Formal Languages, Automata, and Complexity*. Redwood City, CA: Benjamin Cummings, 1989.
- [9] “Internet generated revenue 1996-2002,” 2001. Nua Internet, http://www.nua.ie/surveys/analysis/graphs_charts/comparisons/total_revenue_generated_2002.html.
- [10] “Global internet statistics (by language),” 2001. Global Reach, <http://www.glreach.com/globstats/index.php3>.
- [11] “How many online?,” 2001. Nua Internet, http://www.nua.ie/surveys/how_many_online/index.html.
- [12] A. Kruger, C. L. Giles, F. Coetzee, E. Glover, G. Flake, S. Lawrence, and C. Omlin, “DEAD-LINER: Building a new niche search engine,” in *Ninth International Conference on Information and Knowledge Management, CIKM 2000*, (Washington, DC), pp. 272–281, November 6–11 2000.
- [13] E. J. Glover, G. W. Flake, S. Lawrence, W. P. Birmingham, A. Kruger, C. L. Giles, and D. Pennock, “Improving category specific web search by learning query modifications,” in *Symposium on Applications and the Internet, SAINT*, (San Diego, CA), January 2001.

- [14] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori, "Focused crawling using context graphs," in *26th International Conference on Very Large Databases, VLDB 2000*, (Cairo, Egypt), pp. 527–534, 10–14 September 2000.
- [15] S. Lawrence, "Context in web search," *IEEE Data Engineering Bulletin*, vol. 23, no. 3, pp. 25–32, 2000.
- [16] R. Armstrong, D. Freitag, T. Joachims, and T. Mitchell, "Webwatcher: A learning apprentice for the world wide web," in *AAAI Spring Symposium on Information Gathering from Heterogeneous Distributed Environments*, pp. 6–12, March 1995.
- [17] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," Tech. Rep. RJ 10076, IBM, May 1997.
- [18] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleinberg, "Automatic resource compilation by analyzing hyperlink structure and associated text," in *The Seventh International World Wide Web Conference*, pp. 65–74, 1998.
- [19] "The pagerank citation ranking: Bringing order to the web," January 1998. <http://www-db.stanford.edu/~backrub/pageranksub.ps>.
- [20] N. Kushmerick, D. Weld, and R. Doorenbos, "Wrapper induction for information extraction," in *Proceedings of the International Joint Conference on Artificial Intelligence(IJCAI-97)*, pp. 15–68, 1997.
- [21] S. Soderland, *Learning information extraction rules for semi-structured and free text*. Netherlands: Kluwer Academic Publishers, 1999.
- [22] M. Swain, C. Frankel, and V. Athitsos, "Webseer: An image search engine for the world wide web," in *IEEE Computer Vision and Pattern Recognition Conference (CVPRC)*, June 1997.
- [23] A. Research. Published on the World Wide Web, August 1998. http://www.alexa.com/press/press_releases/webfacts.html.
- [24] J. Challenger, A. Iyengar, and P. Dantzig, "A scalable system for consistently caching dynamic data," in *Proceedings of IEEE INFOCOM '99*, March 1999.
- [25] D. Florescu, A. Levy, and A. Mendelzon, "Database techniques for the world-wide web: A survey," *SIGMOD Record*, vol. 27, no. 3, pp. 59–74, 1998.
- [26] H. Lieberman, N. van Dyke, and A. Vivacqua, "Let's browse: a collaborative web browsing agent," in *Proc. Intl. Conf. on Intelligent User Interfaces*, pp. 65–68, January 1999.
- [27] L. Page and S. Brin, "The anatomy of a large-scale hypertextual web search engine," in *Proceedings of the Seventh International World Wide Web Conference (WWW'98)*, pp. 107–117, April 1998.

- [28] R. Lempel and S. Moran, "The stochastic approach for link-structure analysis (SALSA) and the TKC effect," in *9th International World Wide Web Conference (WWW9)*, pp. 131–160, May 2000.
- [29] A. Borodin, G. O. Roberts, J. S. Rosenthal, and P. Tsaparas, "Finding authorities and hubs from link structures on the world wide web." <http://markov.utstat.toronto.edu/jeff/research.html>, 2000.
- [30] D. Eichmann, "The RBSE spider – balancing effective search against web load," in *Proceedings of the First International Conference on the World Wide Web (WWW'94)*, pp. 113–120, 1994.
- [31] J. Cho, H. Garcia-Molina, and L. Page, "Efficient crawling through URL ordering," in *Proceedings of the 7th World Wide Web Conference (WWW7)*, no. 1–7, pp. 161–172, 1998.
- [32] E. G. Coffman and Z. Liu, "Optimal robot scheduling for web search engines," *Journal of Scheduling*, pp. 15–29, 1997.
- [33] "Inktomi Press Release." Published on the World Wide Web, January 2000. <http://www.inktomi.com/new/press/2000/billion.html>.
- [34] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," in *Seventh International World Wide Web Conference*, (Brisbane, Australia), 1998.
- [35] "Archiving the Internet," 2000. <http://www.archive.org/>.
- [36] "Launch of raging.com," 2000. http://doc.altavista.com/company_info/press/pr050400.html.
- [37] "Google facts page," 2000. <http://www.google.com/corporate/facts.html>.
- [38] P. M. E. De Bra and R. D. J. Post, "Information retrieval in the world-wide web: Making client-based searching feasible," in *Proceedings of the First International Conference on the World Wide Web (WWW'94)*, pp. 183–192, 1994.
- [39] T. Joachims, D. Freitag, and T. Mitchell, "Webwatcher: A tour guide for the World Wide Web," in *Proceedings IJCAI'97*, pp. 770–777, 1997.
- [40] F. Menczer, "Arachnid: Adaptive retrieval agents choosing heuristic neighborhoods for information discovery," in *Proceedings of the 14th International Conference on Machine Learning (ICML'97)*, pp. 227–235, 1997.
- [41] T. M. Mitchell, *Machine Learning*, ch. 8,13, pp. 232–236,367–387. New York: McGraw-Hill, 1997.
- [42] J. Rennie and A. K. McCallum, "Using reinforcement learning to spider the web efficiently," in *Proceedings of the International Conference on Machine Learning (ICML'99)*, pp. 335–343, 1999.

- [43] S. Chakrabarti, M. van den Berg, and B. Dom, "Focused crawling: A new approach to topic-specific web resource discovery," in *Proceedings of the Eighth World Wide Web Conference (WWW8)*, pp. 1623–1640, 1999.
- [44] O. A. McBryan, "Genvl and WWW: Tools for taming the web," in *Proceedings of the 1st Annual International World Wide Web Conference*, pp. 4–11, May 1994.
- [45] J. Boyan, D. Freitag, and T. Joachims, "A machine learning architecture for optimizing web search engines," in *AAAI Workshop on Internet-Based Information Systems*, pp. 334–335, 1994.
- [46] S. Varadarajan and T. ker Chiueh, "SASE: Implementation of a compressed text search engine," in *Usenix Symposium on Internet Technologies and Systems*, 1997.
- [47] D. Beeferman and A. Berger, "Agglomerative clustering of a search engine query log," in *Proceedings of ACM SIGKDD International Conference*, pp. 407–415, 2000.
- [48] S. Lawrence and C. L. Giles, "Searching the World Wide Web," *Science*, vol. 280, no. 5360, pp. 98–100, 1998.
- [49] E. Selberg and O. Etzioni, "Multi-service search and comparison using the metacrawler," in *Proceedings of the 4th World Wide Web Conference (WWW4)*, pp. 195–208, 1995.
- [50] Y. Arens, C. Chee, C. Hsu, and C. Knoblock, "Retrieving and integrating data from multiple information sources," *Journal of Intelligent and Cooperative Information Systems*, vol. 2, pp. 127–158, June 1993.
- [51] D. Dreilinger and A. E. Howe, "Experiences with selecting search engines using meta-search," *ACM Transactions of Information Systems*, pp. 195–222, 1997.
- [52] A. E. Howe and D. Dreilinger, "SavvySearch: A metasearch engine that learns which search engines to query." *AI Magazine*, 1997.
- [53] L. Gravano, H. García-Molina, and A. Tomasic, "Gloss: Text-source discovery over the internet," *ACM Transactions on Database Systems (To Appear)*, vol. 24, June 1999.
- [54] M. Beigi, A. Benitez, and S.-F. Chang, "MetaSEEk: A content-based meta search engine for images," in *SPIE Conference on Storage and Retrieval for Image and Video Database*, (San Jose), pp. 118–128, February 1997. (demo and document:<http://www.ctr.columbia.edu/metaseek>).
- [55] S. Lawrence and C. L. Giles, "Text and image metasearch on the web," in *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA99)*, pp. 829–835, CSREA Press, 1999.
- [56] S. Lawrence and C. L. Giles, "Context and page analysis for improved web search," *IEEE Internet Computing*, vol. 2, no. 4, pp. 38–46, 1998.

- [57] E. J. Glover, S. Lawrence, W. P. Birmingham, and C. L. Giles, "Architecture of a metasearch engine that supports user information needs," in *Eighth International Conference on Information and Knowledge Management (CIKM'99)*, (Kansas City, Missouri), pp. 210–216, ACM Press, November 1999.
- [58] A. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Building domain-specific search engines with machine learning techniques," in *AAAI-99 Spring Symposium on Intelligent Agents in Cyberspace*, 1999.
- [59] J. Connan and C. W. Omlin, "Bibliography field extraction with hidden markov models," tech. rep., University of Stellenbosch, 2000.
- [60] S. R. Lawrence, F. M. Coetzee, E. J. Glover, G. W. Flake, D. M. Pennock, R. Krovetz, F. A. Nielsen, A. F. Kruger, and C. L. Giles, "Persistence of information on the web: Analyzing citations contained in research articles," in *Proceedings of the 9th International Conference on Information Knowledge Management* (A. Agah, ed.), pp. 235–242, 2000.
- [61] K. Bollacker, S. Lawrence, and C. L. Giles, "CiteSeer: An autonomous web agent for automatic retrieval and identification of interesting publications," in *Proceedings of the Second International Conference on Autonomous Agents* (K. P. Sycara and M. Wooldridge, eds.), (New York), pp. 116–123, ACM Press, 1998.
- [62] S. Lawrence, K. Bollacker, and C. L. Giles, "Indexing and retrieval of scientific literature," in *Eighth International Conference on Information and Knowledge Management, CIKM 99*, (Kansas City, Missouri), pp. 139–146, November 1999.
- [63] C. L. Giles, K. Bollacker, and S. Lawrence, "Citeseer: An automatic citation indexing system," in *Digital Libraries 98 - The Third ACM Conference on Digital Libraries* (I. Witten, R. Akscyn, and F. M. S. III, eds.), (Pittsburgh, PA), pp. 89–98, June 1998.
- [64] C. Faloutsos, R. Barber, M. Flickner, W. Niblack, D. Petkovic, and W. Equitz, "Efficient and effective querying by image content," *Journal of Intelligent Information Systems*, vol. 3, pp. 231–262, July 1994.
- [65] S.-F. Chang, W. Chen, H. J. Horace, H. Sundaram, and D. Zhong, "A fully automated content based video search engine supporting spatio-temporal queries," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, pp. 602–615, September 1998.
- [66] S. Eickeler and S. Müller, "Content-based video indexing of tv broadcast news using hidden markov models," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, (Phoenix, USA), pp. 60–65, 1999.
- [67] S. Lawrence, C. L. Giles, and K. Bollacker, "Digital libraries and autonomous citation indexing," *IEEE Computer*, vol. 32, no. 6, pp. 67–71, 1999.

- [68] Y. Yang, "An evaluation of statistical approaches to text categorization," *Journal of Information Retrieval*, pp. 69–90, 1999.
- [69] T. Joachims, "A probabilistic analysis of the rocchio algorithm with tfidf for text categorization," in *Proceedings of the 14th International Conference on Machine Learning (ICML97)*, pp. 143–151, 1997.
- [70] D. G. Roussinov and H. Chen, "A scalable self-organizing map algorithm for textual classification: A neural network approach to thesaurus generation," *Communication Cognition and Artificial Intelligence*, pp. 45–57, 1998.
- [71] D. Koller and M. Sahami, "Hierarchically classifying documents using very few words," in *Proceedings of the 14th International Conference on Machine Learning (ICML97)*, pp. 170–178, 1997.
- [72] "Reverend Thomas Bayes," 2000. <http://www.britannica.com/>.
- [73] D. Heckerman, "Tutorial on learning in Bayesian networks," tech. rep., Microsoft Research, July 1995.
- [74] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," in *AAAI-98 Workshop on Learning for Text Categorization*, 1998.
- [75] E. Wiener, J. O. Pedersen, and A. S. Weigend, "A neural network approach to topic spotting," in *the 4th Annual Symposium on Document Analysis and Information Retrieval (SDAIR'95)*, (Las Vegas, Nevada, USA), pp. 317–332, April 1995.
- [76] D. Mladenic, "Text-learning and related intelligent agents," *IEEE Expert Special Issue on Application of Intelligent Information Retrieval*, pp. 44–54, July-August 1999.
- [77] S. Haykin, *Neural Networks A comprehensive Foundation*, ch. 6. Prentice Hall, Upper Saddle River, New Jersey 07458: Tom Robbins, 2 ed., 1999.
- [78] J. T.-Y. Kwok, "Automated text categorization using support vector machines," in *Proceedings of the International Conference on Neural Information Processing ICONIP*, (Kitakyushu, Japan), pp. 347–351, October 1998.
- [79] S. Haykin, *Neural Networks A comprehensive Foundation*, ch. 2, p. 94. Prentice Hall, Upper Saddle River, New Jersey 07458: Tom Robbins, 2 ed., 1999.
- [80] C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala, "Latent semantic indexing: A probabilistic analysis," in *Proceedings of the ACM Conference on Principles of Database Systems (PODS)*, (Seattle), pp. 159–168, 1998.
- [81] D. Miller, T. Leek, and R. Schwartz, "BBN at TREC-7: Using hidden Markov models for information retrieval," in *The Seventh Text Retrieval Conference, TREC-7. NIST Special Publications*, pp. 80–89, 1999.

- [82] T. R. Leek, "Information extraction using hidden Markov models." Master's thesis, UC San Diego, 1997.
- [83] L. Firoiu, T. Oates, and P. Cohen, "Learning regular languages from positive evidence," in *Twentieth Annual Meeting of the Cognitive Science Society*, pp. 350–355, 1998.
- [84] C. Hsu, "Initial results on wrapping semistructured web pages with finite-state transducers and contextual rules," in *The 1998 Workshop on AI and Information Integration*, (Madison), pp. 66–73, AAAI Press, 1998.
- [85] L. Liu, C. Pu, and W. Han, "Xwrap: An XML-enabled wrapper construction system for web information sources," in *International Conference on Data Engineering (ICDE)*, pp. 611–621, 2000.
- [86] I. Muslea, S. Minton, and C. Knoblock, "Stalker: Learning extraction rules for semistructured, web-based information," in *Workshop on AI and Information Integration (AAAI-98)*, (Menlo Park, CA), AAAI Press, 1998.
- [87] E. Riloff and W. Lehnert, "Information extraction as a basis for high-precision text classification," *ACM Transactions on Information Systems*, vol. 12, no. 3, pp. 296–333, 1994.
- [88] J. P. Egan, *Signal Detection Theory and ROC Analysis*, ch. 1–2, 6. Academic Press, New York: Academic Press, 1975.
- [89] K. R. Sharma, F. Rotta, J. Romano, and D. R. Ayyar, "Early diagnosis of carpal tunnel syndrome: Comparison of digit 1 with wrist and distoproximal ratio," *Neurology and Clinical Neurophysiology*, March 2001.
- [90] B. W. Lindgren, *STATISTICAL THEORY*, ch. 6, pp. 308–314. New York: MacMillan, 2 ed., 1968.
- [91] D. M. Green and J. Swets, *Signal Detection Theory and Psychophysics*, pp. 7–29. New York: Krieger Publishing Co., 1974.
- [92] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in kernel methods - support vector learning* (B. Schölkopf, C. Burges, and A. Smola, eds.), MIT Press, 1998. <http://www.research.microsoft.com/~jplatt/>.
- [93] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," Tech. Rep. 98-14, Microsoft Research, Redmond, Washington, April 1998. <http://www.research.microsoft.com/~jplatt/>.
- [94] J. Platt, "Using sparseness and analytic qp to speed training of support vector machines," in *Advances in Neural Information Processing Systems* (M. S. Kearns, S. A. Solla, and D. A. Cohn, eds.), MIT Press, 1999. <http://www.research.microsoft.com/~jplatt/>.

- [95] E. J. Dudewicz and S. N. Mishra, *Modern Mathematical Statistics*. New York: John Wiley, 1988.
- [96] F. Coetzee, S. Lawrence, and C. L. Giles, "Bayesian classification and feature selection from finite data sets," in *Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI-2000)*, (Stanford, CA), pp. 89–97, 2000.